NOTICE

The information in this document has been carefully checked and is believed to be entirely reliable. FORCE COMPUTERS makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. FORCE COMPUTERS reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

FORCE COMPUTERS assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of FORCE COMPUTERS GmbH/Inc.

FORCE COMPUTERS does not convey to the purchaser of the product described herein any license under the patent rights of FORCE COMPUTERS GmbH/Inc. nor the rights of others.

FORCE COMPUTERS strives to contribute to the safety and preservation of the environment with the same care it applies to the performance and quality of its products.

To support this effort, this Product Manual has been printed on paper that is completely chlorine free and conservation friendly.
List of Tables

Table 1. Specifications of the SPARC CPU-5V .......................................................... 4
Table 2. Ordering Information ................................................................................ 6
Table 3. History of Manual .................................................................................... 8
Table 4. Default Switch Settings ............................................................................. 12
Table 5. Device Alias Definitions .......................................................................... 22
Table 6. Setting Configuration Parameters ............................................................ 23
Table 7. Diagnostic Routines .................................................................................. 25
Table 8. Commands to Display System Information .............................................. 29
Table 9. Front Panel Layout .................................................................................. 33
Table 10. SPARC CPU-5V Connectors .................................................................. 34
Table 11. Ethernet Connector Pinout ..................................................................... 35
Table 12. Serial Port A and B Connector Pinout ..................................................... 36
Table 13. SCSI 50-Pin Connector ........................................................................... 38
Table 14. Keyboard/Mouse Connector Pinout ........................................................ 40
Table 15. VME P2 Connector Pinout ...................................................................... 41
Table 16. IOBP-10 P1 Pinout ................................................................................ 43
Table 17. IOBP-10 P2 Pinout (SCSI) ..................................................................... 45
Table 18. IOBP-10 P3 Pinout (Floppy) .................................................................. 46
Table 19. IOBP-10 P4 Pinout (Centronics) ............................................................. 47
Table 20. IOBP-10 P5 Pinout (Serial) ..................................................................... 48
Table 21. IOBP-10 P6 Pinout (Ethernet) ................................................................. 48
Table 22. Physical Memory Map of microSPARC-II .............................................. 54
Table 23. Bank Selection ....................................................................................... 55
Table 24. CPU-5V Memory Banks ......................................................................... 56
Table 25. MEM-5 Memory Banks ......................................................................... 56
Table 26. Physical Memory Map of SBus on SPARC CPU-5V ............................... 57
Table 27. NCR89C105 Chip Address Map ............................................................ 63
Table 28. RS-232, RS-422 or RS-485 Configuration .............................................. 64
Table 29. Serial Ports A and B Pinout List (RS-232) ............................................... 65
Table 30. Switch Settings for Ports A and B (RS-232) ............................................. 65
Table 31. Serial Ports A and B Pinout List (RS-422) .............................................. 66
Table 32. Switch Settings for Ports A and B (RS-422) ............................................. 67
Table 33. Serial Ports A and B Pinout List (RS-485) .............................................. 68
Table 34. Switch Settings for Ports A and B (RS-485) ............................................. 68
Table 35. 8-Bit Local I/O Devices ......................................................................... 70
Table 36. Boot Flash Memory Capacity .................................................................. 71
Table 37. User Flash Memory Capacity .................................................................. 72
Table 38. Physical Memory Map of VMEbus Interface on SPARC CPU-5V .......... 79
Table 39. Front Panel Layout ................................................................................ 83
Table 40. Interrupt Mapping .................................................................................. 109
Table 41. VMEbus Transaction Timer Timeout Values .......................................... 153
Table 42. Watchdog Timer Timeout Values ............................................................. 156
This page intentionally left blank
SECTION 1 INTRODUCTION .......................................................................................................................... 1

1. Getting Started ........................................................................................................................................... 1

1.1. The SPARC CPU-5V Technical Reference Manual Set ........................................................................ 1

1.2. Summary of the SPARC CPU-5V .......................................................................................................... 2

1.3. Specifications ........................................................................................................................................ 4

1.3.1. Ordering Information ......................................................................................................................... 6

1.4. History of the Manual ........................................................................................................................... 8

SECTION 2 INSTALLATION ............................................................................................................................ 9

2. Introduction ................................................................................................................................................ 9

2.1. Caution .................................................................................................................................................. 9

2.2. Location Diagram of the SPARC CPU-5V Board .................................................................................. 9

2.3. Before Powering Up ............................................................................................................................... 12

2.3.1. Default Switch Settings ..................................................................................................................... 12

2.4. Powering Up ......................................................................................................................................... 14

2.4.1. VME Slot-1 Device ............................................................................................................................ 14

2.4.2. VMEbus SYSRESET Enable/Disable .............................................................................................. 15

2.4.2.1. SYSRESET Input ........................................................................................................................... 15

2.4.2.2. SYSRESET Output ......................................................................................................................... 15

2.4.3. Serial Ports ....................................................................................................................................... 15

2.4.4. RESET and ABORT Key Enable ....................................................................................................... 16

2.4.5. SCSI Termination .............................................................................................................................. 16

2.4.5.1. SCSI Termination at the Front Panel .......................................................................................... 16

2.4.5.2. SCSI Termination at P2 ................................................................................................................ 16

2.4.6. Boot Flash Memory Write Protection ............................................................................................. 17

2.4.7. User Flash Memory Write Protection ............................................................................................. 17

2.4.8. Reserved Switches ............................................................................................................................ 17

2.4.9. Parallel Port or Floppy Interface via VME P2 Connector ................................................................ 18

2.4.10. Ethernet via Front Panel or VME P2 Connector ............................................................................ 19

2.5. OpenBoot Firmware ............................................................................................................................... 20

2.5.1. Boot the System ............................................................................................................................... 20

2.5.2. NVRAM Boot Parameters ................................................................................................................ 23

2.5.3. Diagnostics ....................................................................................................................................... 24

2.5.4. Display System Information ........................................................................................................... 28

2.5.5. Reset the System ............................................................................................................................... 30

2.5.6. OpenBoot Help ................................................................................................................................. 30

2.6. Front Panel .......................................................................................................................................... 32

2.6.1. Features of the Front Panel .............................................................................................................. 33

2.7. SPARC CPU-5V Connectors ................................................................................................................ 34

2.7.1. Ethernet Connector Pinout ................................................................................................................ 35

2.7.2. Serial Port A and B Connector Pinout ............................................................................................ 36

2.7.3. SCSI Connector Pinout .................................................................................................................... 38
SECTION 3 HARDWARE DESCRIPTION ........................................................................51

3. Overview .............................................................................................................. 51
3.1. Block Diagram .................................................................................................... 51
3.2. The microSPARC-II Processor ........................................................................ 53
3.2.1. Features of the microSPARC-II Processor .................................................... 53
3.2.2. Address Mapping for microSPARC-II .......................................................... 54
3.3. The Shared Memory .......................................................................................... 55
3.4. Memory Module MEM-5.................................................................................. 56
3.5. SBus Participants .............................................................................................. 57
3.5.1. Address Mapping for SBus Slots on the SPARC CPU-5V ............................... 57
3.6. NCR89C100 (MACIO) ...................................................................................... 58
3.6.1. Features of the NCR89C100 on the SPARC CPU-5V .................................... 59
3.6.2. SCSI .............................................................................................................. 60
3.6.3. SCSI Termination ....................................................................................... 60
3.6.4. Ethernet ...................................................................................................... 61
3.6.5. Parallel Port .................................................................................................. 61
3.7. NCR89C105 (SLAVIO) ...................................................................................... 62
3.7.1. Features of the NCR89C105 on the SPARC CPU-5V .................................... 62
3.7.2. Address Map of Local I/O Devices on SPARC CPU-5V ............................... 63
3.7.3. Serial I/O Ports ............................................................................................ 64
3.7.4. RS-232, RS-422 or RS-485 Configuration .................................................... 64
3.7.5. RS-232 Hardware Configuration ................................................................ 65
3.7.6. RS-422 Hardware Configuration ................................................................ 66
3.7.7. RS-485 Hardware Configuration ................................................................ 68
3.7.8. Keyboard and Mouse Port .......................................................................... 69
3.7.9. Floppy Disk Interface .................................................................................. 69
3.7.10. 8-Bit Local I/O Devices ............................................................................. 70
3.7.11. Boot Flash Memory .................................................................................... 71
3.7.12. User Flash Memory ..................................................................................... 72
3.7.13. Programming the On-board Flash Memories .............................................. 73
3.7.13.1. Flash Memory Programming Voltage Control Register ......................... 74
3.7.13.2. Flash Memory Programming Control Register 1 ................................... 75
3.7.13.3. Flash Memory Programming Control Register 2 ................................... 76
3.7.14. RTC/NVRAM ............................................................................................ 77
3.8. VMEbus Interface ............................................................................................ 78
3.8.1. Address Mapping for the VMEbus Interface FGA-5000 ............................... 79
3.8.2. Adaptation of the FGA-5000 ....................................................................... 80
3.8.3. VMEbus SYSRESET Enable/Disable ............................................................ 81
# Table of Contents


<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8.3.1.</td>
<td>SYSRESET Input</td>
</tr>
<tr>
<td>3.8.3.2.</td>
<td>SYSRESET Output</td>
</tr>
<tr>
<td>3.9.</td>
<td>On-board Control Registers (System Configuration)</td>
</tr>
<tr>
<td>3.10.</td>
<td>Front Panel</td>
</tr>
<tr>
<td>3.10.1.</td>
<td>RESET and ABORT Keys</td>
</tr>
<tr>
<td>3.10.1.1.</td>
<td>The RESET Key</td>
</tr>
<tr>
<td>3.10.1.2.</td>
<td>The ABORT Key</td>
</tr>
<tr>
<td>3.10.2.</td>
<td>Front Panel Status LEDs</td>
</tr>
<tr>
<td>3.10.2.1.</td>
<td>USER LED 1 Control Register</td>
</tr>
<tr>
<td>3.10.2.2.</td>
<td>USER LED 2 Control Register</td>
</tr>
<tr>
<td>3.10.3.</td>
<td>Diagnostic LED (Hex Display)</td>
</tr>
<tr>
<td>3.10.3.1.</td>
<td>Seven Segment LED Display Control Register</td>
</tr>
<tr>
<td>3.10.4.</td>
<td>Rotary Switch</td>
</tr>
<tr>
<td>3.10.4.1.</td>
<td>Rotary Switch Status Register</td>
</tr>
<tr>
<td>3.11.</td>
<td>Additional Registers</td>
</tr>
<tr>
<td>3.11.1.</td>
<td>FMB Channel 0 Data Discard Status Register</td>
</tr>
<tr>
<td>3.11.2.</td>
<td>FMB Channel 1 Data Discard Status Register</td>
</tr>
</tbody>
</table>

## SECTION 4 OpenBoot

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>Software</td>
</tr>
<tr>
<td>4.1.</td>
<td>OpenBoot</td>
</tr>
<tr>
<td>4.2.</td>
<td>VMEbus Interface</td>
</tr>
<tr>
<td>4.2.1.</td>
<td>Generic Information</td>
</tr>
<tr>
<td>4.2.2.</td>
<td>Register Addresses</td>
</tr>
<tr>
<td>4.2.3.</td>
<td>Register Accesses</td>
</tr>
<tr>
<td>4.2.4.</td>
<td>VMEbus Interrupt Handler</td>
</tr>
<tr>
<td>4.2.5.</td>
<td>VMEbus Arbiter</td>
</tr>
<tr>
<td>4.2.6.</td>
<td>VMEbus Requester</td>
</tr>
<tr>
<td>4.2.7.</td>
<td>VMEbus Status Signals</td>
</tr>
<tr>
<td>4.2.8.</td>
<td>VMEbus Master Interface</td>
</tr>
<tr>
<td>4.2.9.</td>
<td>VMEbus Slave Interface</td>
</tr>
<tr>
<td>4.2.10.</td>
<td>VMEbus Device Node</td>
</tr>
<tr>
<td>4.2.11.</td>
<td>VMEbus NVRAM Configuration Parameters</td>
</tr>
<tr>
<td>4.2.12.</td>
<td>DMA Controller Support</td>
</tr>
<tr>
<td>4.2.13.</td>
<td>Mailboxes and Semaphores</td>
</tr>
<tr>
<td>4.2.14.</td>
<td>FORCE Message Broadcast</td>
</tr>
<tr>
<td>4.2.15.</td>
<td>Diagnostic</td>
</tr>
<tr>
<td>4.2.16.</td>
<td>Miscellanea</td>
</tr>
<tr>
<td>4.3.</td>
<td>Standard Initialization of the VMEbus Interface</td>
</tr>
<tr>
<td>4.3.1.</td>
<td>SPARC FGA-5000 Registers</td>
</tr>
<tr>
<td>4.3.2.</td>
<td>VMEbus Transaction Timer</td>
</tr>
<tr>
<td>4.3.3.</td>
<td>SBus Rerun Limit</td>
</tr>
<tr>
<td>4.3.4.</td>
<td>Interrupts</td>
</tr>
</tbody>
</table>
Table of Contents

4.3.5. SBUS Slot 5 Address Map ................................................................. 155
4.4. System Configuration ................................................................. 156
4.4.1. Watchdog Timer ............................................................................. 156
4.4.2. Watchdog Timer NVRAM Configuration Parameters ................. 158
4.4.3. Abort Switch ................................................................................ 158
4.4.4. Abort Switch NVRAM Configuration Parameter ...................... 159
4.4.5. LEDs, Seven-Segment Display and Rotary Switch ..................... 159
4.4.6. Reset ............................................................................................. 160
4.5. Flash Memory Support ...................................................................... 162
4.5.1. Flash Memory Programming ....................................................... 162
4.5.2. Flash Memory Device ..................................................................... 164
4.5.3. Loading and Executing Programs from USER Flash Memory .......... 166
4.5.4. Controlling the Flash Memory Interface ........................................ 167
4.6. Onboard Interrupts ............................................................................. 169
4.6.1. VMEbus Interrupts .......................................................................... 169
4.6.2. SYSFAIL Interrupt ......................................................................... 170
4.6.3. ACFAIL Interrupt ........................................................................... 171
4.6.4. ABORT Interrupt ............................................................................ 172
4.6.5. Watchdog Timer Interrupt ............................................................ 172

SECTION 5 CIRCUIT SCHEMATICS ........................................................... 175
5. CPU-5V Schematics ............................................................................ 175
5.1. MEM-5 Schematics .......................................................................... 176

SECTION 6 SUN OPEN BOOT DOCUMENTATION .................................... 177
6. Insert your OPEN BOOT 2.0 PROM MANUAL SET here ..................... 177

SECTION 7 APPENDIX .............................................................................. 179
7. Product Error Report ........................................................................... 179

SECTION 8 USER’S NOTES ..................................................................... 181
8. User’s Notes ....................................................................................... 181

SECTION 9 OPTIONS .............................................................................. 183
9. Additional Options .............................................................................. 183
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECTION 10 MODIFICATIONS</td>
<td>Additional Modifications</td>
<td>185</td>
</tr>
<tr>
<td>SECTION 11 APPLICATIONS</td>
<td>Additional Applications</td>
<td>187</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1. Block Diagram of the SPARC CPU-5V ................................................................. 3
Figure 2. Diagram of the CPU-5V (Top View) ................................................................. 10
Figure 3. Diagram of the CPU-5V (Bottom View) ......................................................... 11
Figure 4. Floppy Interface Via VME P2 Connector ......................................................... 18
Figure 5. Ethernet Interface via Front Panel ................................................................. 19
Figure 6. Diagram of the Front Panel ............................................................................ 32
Figure 7. Pinout of the Ethernet Cable Connector ....................................................... 35
Figure 8. Serial Ports A and B Connector Pinout ......................................................... 37
Figure 9. Pinout of SCSI Connector ............................................................................. 39
Figure 10. Keyboard/Mouse Connector ....................................................................... 40
Figure 11. The IOBP-10 ............................................................................................... 42
Figure 12. Block Diagram of the SPARC CPU-5V ....................................................... 52
Figure 13. Segments of the Hex Display ..................................................................... 88
SECTION 1

1. Getting Started

This SPARC CPU-5V Technical Reference Manual provides a comprehensive guide to the SPARC CPU-5V board you purchased from FORCE COMPUTERS. In addition, each board delivered by FORCE includes an Installation Guide.

Please take a moment to examine the Table of Contents of the SPARC CPU-5V Technical Reference Manual to see how this documentation is structured. This will be of value to you when looking for information in the future.

1.1 The SPARC CPU-5V Technical Reference Manual Set

When purchased from FORCE, this set includes the SPARC CPU-5V Technical Reference Manual as well as two additional books. These two books are listed here:

Set of Data Sheets for the SPARC CPU-5V

OPEN BOOT PROM 2.0 MANUAL SET

The Set of Data Sheets for the SPARC CPU-5V contains the following data sheets.

- NCR SBus I/O Chipset Data Manual
- microSPARC-II User’s Manual (STP1012PGA)
- SGS-THOMSON MK48T08(B)-10/12/15/20
- AMD Flash EPROM (AM28F020)
- Intel Flash Memory (28F008SA-L)

The OPEN BOOT PROM 2.0 MANUAL SET contains the following three sections.

- Open Boot 2.0 Quick Reference
- Open Boot 2.0 Command Reference
- FCODE Programs
1.2 Summary of the SPARC CPU-5V

The SPARC CPU-5V addresses embedded applications where processing performance is as important as VMEbus throughput. Based on FORCE COMPUTERS FGA-5000 VMEbus to SBus interface gate array, the SPARC CPU-5V provides high speed VMEbus transfer capabilities for standard transfers and extended 64-bit MBLT transfers. In addition, the SPARC CPU-5V implements the capabilities of Sun Microsystems' SPARCstation 5 workstation on a single-slot VMEbus board.

The SPARC CPU-5V is powered by the microSPARC-II processor, which delivers a sustained processing performance of 76 SPECint92 and 65 SPECfp92. The complete suite of I/O functions includes fast SCSI-2, Ethernet, floppy disk, serial I/O, Centronics parallel I/O and keyboard/mouse ports making the SPARC CPU-5V the ideal solution for computing and VME transfer intensive embedded applications.

A complete 64-bit VMEbus interface and two standard SBus slots enable the expansion of I/O memory and processing performance with a broad range of off-the-shelf solutions. The software support for the SPARC CPU-5V ranges from Solaris, the most popular implementation of the UNIX operating system on a RISC architecture, to sophisticated real-time operating systems such as VxWorks.

The SPARC CPU-5V is a single board computer combining workstation performance and functionality with the ruggedness and expandability of an industry-standard single-slot 6U VMEbus board.
FIGURE 1. Block Diagram of the SPARC CPU-5V
1.3 Specifications

Below is a table outlining the specifications of the SPARC CPU-5V board.

**Table 1: Specifications of the SPARC CPU-5V**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Specification Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>85 or 110 MHz microSPARC-II 64.0 / 76 SPECint92 54.6 / 65 SPECfp92</td>
</tr>
<tr>
<td>Memory Management Unit</td>
<td>SPARC Reference MMU</td>
</tr>
<tr>
<td>Data/Instruction Cache</td>
<td>8 Kbyte/16 Kbyte</td>
</tr>
<tr>
<td>Main Memory</td>
<td>16-or 64-Mbyte base board DRAM, expandable to 192 MB with mezzanine modules</td>
</tr>
<tr>
<td>SBus Slots</td>
<td>2</td>
</tr>
<tr>
<td>SCSI-2 with DMA to SBus</td>
<td>10 Mbytes/sec fast SCSI-2 I/O on front panel and P2</td>
</tr>
<tr>
<td>Ethernet with DMA to SBus</td>
<td>10 Mbits/sec, AM7990 compatible AUI port on front panel or P2</td>
</tr>
<tr>
<td>Parallel Port with DMA to SBus</td>
<td>3.4 Mbytes/sec I/O on P2 via switch matrix</td>
</tr>
<tr>
<td>Floppy Disk Interface</td>
<td>250, 300, 500 Kbytes/sec and 1 Mbyte/sec I/O on P2 via switch matrix</td>
</tr>
<tr>
<td>Serial I/O</td>
<td>2 RS-232 ports, RS-422/485 option via hybrid modules, I/O on front panel and P2</td>
</tr>
<tr>
<td>Keyboard/Mouse Port</td>
<td>Sun compatible, on front panel and P2</td>
</tr>
<tr>
<td>Counters/Timers</td>
<td>Two 22-bit, 500 ns resolution</td>
</tr>
<tr>
<td>Boot Flash Memory</td>
<td>512 Kbyte, on-board programmable Hardware write protection</td>
</tr>
<tr>
<td>User Flash Memory</td>
<td>2 Mbytes, on-board programmable Hardware write protection</td>
</tr>
<tr>
<td>RTC/NVRAM/Battery</td>
<td>M48T08</td>
</tr>
</tbody>
</table>
Table 1: Specifications of the SPARC CPU-5V (Continued)

<table>
<thead>
<tr>
<th>VMEbus Interface</th>
<th>64-bit master/slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>A32, A24, A16</td>
</tr>
<tr>
<td></td>
<td>D64, D32, D16, D8</td>
</tr>
<tr>
<td></td>
<td>MBLT, BLT</td>
</tr>
<tr>
<td>Slave</td>
<td>A32, A24, A16</td>
</tr>
<tr>
<td></td>
<td>D64, D32, D16, D8</td>
</tr>
<tr>
<td></td>
<td>MBLT, BLT, UAT</td>
</tr>
<tr>
<td>Additional Features</td>
<td>Reset and Abort switches</td>
</tr>
<tr>
<td></td>
<td>Status LEDs, HEX display, Rotary switch, Power-on reset circuitry, Voltage sensor</td>
</tr>
<tr>
<td>Firmware</td>
<td>OpenBoot with diagnostics</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>+5V</td>
</tr>
<tr>
<td>(no SBus Modules installed)</td>
<td>5.0 A</td>
</tr>
<tr>
<td></td>
<td>+12V / -12V</td>
</tr>
<tr>
<td></td>
<td>0.7 / 0.2A</td>
</tr>
<tr>
<td>Environmental Conditions</td>
<td>0°C to +55°C</td>
</tr>
<tr>
<td>Temperature (Operating)</td>
<td>-40°C to +85°C</td>
</tr>
<tr>
<td>Temperature (Storage)</td>
<td>0% to 95% noncondensing</td>
</tr>
<tr>
<td>Humidity</td>
<td></td>
</tr>
<tr>
<td>Board Size</td>
<td>Single-Slot 6U form factor</td>
</tr>
<tr>
<td></td>
<td>160.00 x 233.35 mm</td>
</tr>
<tr>
<td></td>
<td>6.29 x 9.18 inches</td>
</tr>
</tbody>
</table>
### 1.3.1 Ordering Information

This next page contains a list of the product names and their descriptions.

#### Table 2: Ordering Information

<table>
<thead>
<tr>
<th>Catalog Name</th>
<th>Product Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-5V/16-110-2</td>
<td>110 MHz microSPARC-II CPU board with 16-Mbyte base board DRAM, 2-Mbyte User Flash Memory, SCSI, Ethernet, floppy disk, parallel and 2 serial I/O ports, 64-bit VMEbus interface, 2 SBus slots, OpenBoot firmware. Installation guide included.</td>
</tr>
<tr>
<td>CPU-5V/16-85-2</td>
<td>as above, except 85 MHz microSPARC-II.</td>
</tr>
<tr>
<td>CPU-5V/64-110-2</td>
<td>as above, except 110 MHz microSPARC-II and 64-Mbyte base board DRAM.</td>
</tr>
<tr>
<td>CPU-5V/64-85-2</td>
<td>as above, except 85 MHz microSPARC-II.</td>
</tr>
<tr>
<td>MEM-5/16</td>
<td>16-Mbyte mezzanine memory module for use on the SPARC CPU-5V. Up to two memory modules can be used.</td>
</tr>
<tr>
<td>MEM-5/64</td>
<td>64-Mbyte mezzanine memory module for use on the SPARC CPU-5V. Up to two memory modules can be used.</td>
</tr>
<tr>
<td>SBus Modules</td>
<td></td>
</tr>
<tr>
<td>SBus/GX</td>
<td>Color 2-D and 3-D wireframe accelerator 1152x900, 8 bits per pixel, single SBus slot.</td>
</tr>
<tr>
<td>SBus/TGX</td>
<td>Color 2-D and 3-D wireframe high performance graphics accelerator up to 1152x900, 1-Mbyte VRAM, 8 bits per pixel, single SBus slot.</td>
</tr>
<tr>
<td>SBus/TGX+</td>
<td>Color 2-D and 3-D wireframe high performance graphics accelerator up to 1600x1280, 4-Mbyte VRAM, 8 bits per pixel, double buffering, single SBus slot.</td>
</tr>
<tr>
<td>SBus/FP</td>
<td>6U front panel for up to 2 SBus cards.</td>
</tr>
<tr>
<td>Accessories</td>
<td></td>
</tr>
<tr>
<td>IOBP-10</td>
<td>I/O backpanel on VMEbus P2 with flat cable connectors for Ethernet, SCSI, serial I/O and parallel/floppy disk interface for use with the CPU-5V.</td>
</tr>
<tr>
<td>Serial-2CE</td>
<td>Serial adapter cable 26-pin micro D-Sub to 25-pin D-Sub for use with the CPU-5V.</td>
</tr>
<tr>
<td>Catalog Name</td>
<td>Product Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>FH003/SET</td>
<td>Hybrid modules for RS-422 serial I/O configuration.</td>
</tr>
<tr>
<td>FH005/SET</td>
<td>Hybrid modules for RS-485 serial I/O configuration.</td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
<tr>
<td>Solaris 2.x/CPU-5V</td>
<td>Solaris 2.x package with Desktop Right-To-Use license, VMEbus driver on tape. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>Solaris 2.x/Client-RTU</td>
<td>Solaris 2.x Desktop Right-To-Use license. Without media. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>Solaris 2.x/Server-RTU-up</td>
<td>Solaris 2.x Desktop to Workgroup Server Right-To-Use upgrade license. Without media. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>Solaris 2.x/UM</td>
<td>Solaris 2.x operating system user manual. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>Solaris 1.x/CPU-5V</td>
<td>Solaris 1.x package with Right-To-Use license, VMEbus driver on tape. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>Solaris 1.x/RTU-up</td>
<td>Solaris 1.x Right-To-Use license. Without media. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>Solaris 1.x/UM</td>
<td>Solaris 1.x multiuser Right-To-Use license. Without media. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>Solaris 1.x/OF-RTU</td>
<td>Solaris 1.1 operating system user manual. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>VxWorks/DEV SPARC products</td>
<td>VxWorks development package for SPARC host and target. Please contact your local sales representative for current version information.</td>
</tr>
<tr>
<td>VxWorks/BSP CPU-5V</td>
<td>VxWorks board support package for CPU-5V. Please contact your local sales representative for current version information.</td>
</tr>
</tbody>
</table>
1.4 History of the Manual

Below is a description of the publication history of this *SPARC CPU-5V Technical Reference Manual*.

Table 3: History of Manual

<table>
<thead>
<tr>
<th>Revision No.</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First Print</td>
<td>April 1995</td>
</tr>
</tbody>
</table>
SECTION 2

2. Introduction

This Installation Section provides guidelines for powering up the SPARC CPU-5V board. The Installation Section, which you have in your hand now, appears both as Section 2 of the SPARC CPU-5V Technical Reference Manual and as a stand-alone Installation Guide. This stand-alone Installation Guide is delivered by FORCE COMPUTERS with every board. The SPARC CPU-5V Technical Reference Manual provides a comprehensive hardware and software guide to your board and is intended for those persons who require complete information.

2.1 Caution

Please read this Installation Section before installing the board. Take a moment to examine the Table of Contents to see how this documentation is structured. This will be of value to you when looking for specific information in the future.

2.2 Location Diagram of the SPARC CPU-5V Board

A location diagram showing the important components on the top side of the CPU-5V appears on the next page. On the page next to it, there is a location diagram showing the bottom side of the CPU-5V. Both of these diagrams show only the components on the board which are of interest to the user.
FIGURE 2. Diagram of the CPU-5V (Top View)
FIGURE 3.  Diagram of the CPU-5V (Bottom View)
2.3 Before Powering Up

Before powering up, please make sure that the default switch settings are all set according to the table below. Check these switch settings before powering up the SPARC CPU-5V because the board is configured for power up according to these default settings. For the position of the switches on the board, please see “Diagram of the CPU-5V (Top View)” on page 10.

2.3.1 Default Switch Settings

Table 4: Default Switch Settings

<table>
<thead>
<tr>
<th>Diagram of Switch with Default Setting</th>
<th>Switches</th>
<th>Default Setting</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCH 4 (Serial A Configuration)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SW4-1</td>
<td>ON</td>
<td>On = SER_TRXCA to TXC_A_CONN (Pin 24)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Off = SER_KTS_A to TXC_A_CONN (Pin 24)</td>
</tr>
<tr>
<td></td>
<td>SW4-2</td>
<td>OFF</td>
<td>On = CTS_A_CONN (Pin 5) to SER_RXC and \</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pullup to SER_CTSA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Off = RTX_C_A_CONN (Pin 17) to SER_RXC and \</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTS_A_CONN (Pin 5) to SER_CTSA</td>
</tr>
<tr>
<td></td>
<td>SW4-3</td>
<td>OFF</td>
<td>On = SER_TRXCA to RTS_A_CONN (Pin 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Off = SER_KTS_A to RTS_A_CONN (Pin 4)</td>
</tr>
<tr>
<td></td>
<td>SW4-4</td>
<td>OFF</td>
<td>RESERVED</td>
</tr>
<tr>
<td>SWITCH SW5 (Controls Serial Channel B)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SW5-1</td>
<td>ON</td>
<td>On = SER_TRXCB to TXC_B_CONN (Pin 25)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Off = SER_KTS_B to TXC_B_CONN (Pin 25)</td>
</tr>
<tr>
<td></td>
<td>SW5-2</td>
<td>OFF</td>
<td>On = CTS_B_CONN (Pin 13) to SER_RXC and \</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pullup to SER_CTSB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Off = RTX_C_B_CONN (Pin 22) to SER_RXC and \</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTS_B_CONN (Pin 13) to SER_CTSA</td>
</tr>
<tr>
<td></td>
<td>SW5-3</td>
<td>OFF</td>
<td>On = SER_TRXCB to RTS_B_CONN (Pin 19)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Off = SER_KTS_B to RTS_B_CONN (Pin 19)</td>
</tr>
<tr>
<td></td>
<td>SW5-4</td>
<td>OFF</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>
### Table 4: Default Switch Settings (Continued)

<table>
<thead>
<tr>
<th>Diagram of Switch with Default Setting</th>
<th>Switches</th>
<th>Default Setting</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SWITCH 6</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| SW6                                    | SW6-1    | OFF            | On = SCSI-Term Front Panel disabled  
                                         |          |                | Off = SCSI-Term Front Panel automatic |
|                                        | SW6-2    | OFF            | On = SCSI-Term VME P2 disabled       
                                         |          |                | Off = SCSI-Term VME P2 enabled       |
|                                        | SW6-3    | OFF            | On = Write Boot Flash enabled        
                                         |          |                | Off = Write Boot Flash disabled      |
|                                        | SW6-4    | OFF            | On = Write User Flash enabled        
                                         |          |                | Off = Write User Flash disabled      |
| **SWITCH 7**                           |          |                |          |
| SW7                                    | SW7-1    | ON             | On = RESET Switch enabled            
                                         |          |                | Off = RESET Switch disabled          |
|                                        | SW7-2    | ON             | On = ABORT Switch enabled            
                                         |          |                | Off = ABORT Switch disabled          |
|                                        | SW7-3    | ON             | On = VME_SYSRESET input enabled      
                                         |          |                | Off = VME_SYSRESET input disabled    |
|                                        | SW7-4    | ON             | On = VME_SYSRESET output enabled     
                                         |          |                | Off = VME_SYSRESET output disabled   |
| **SWITCH 8**                           |          |                |          |
| SW8                                    | SW8-1    | OFF            | On = LCA Configuration Mode Download 
                                         |          |                | Off = LCA Configuration Mode Serial PROM |
|                                        | SW8-2    | ON             | On = VME Slot 1 Function enabled     
                                         |          |                | Off = VME Slot 1 Function disabled   |
|                                        | SW8-3    | OFF            | On = Power Sense 4.5V                
                                         |          |                | Off = Power Sense 4.75V              |
|                                        | SW8-4    | OFF            | Reserved                             |
2.4 Powering Up

The initial power up can easily be done by connecting a terminal to ttya (serial port A). The advantage of using a terminal is that no frame buffer, monitor, or keyboard is used for initial power up, which facilitates a simple start up.

Please see the chapter “OpenBoot Firmware” on page 20 for more detailed information on booting the system.

2.4.1 VME Slot-1 Device

The SPARC CPU-5V can be plugged into any VMEbus slot; however, the default configuration sets the board as a VME slot-1 device, which functions as VME system controller. To configure your CPU-5V in order that it is not a VME slot-1 device, the default configuration must be changed so that SW8-2 is OFF. In that case, it would also be necessary to change the SW7-4 to OFF, so that the VME_SYSRESET output is disabled.

CAUTION

Before installing the SPARC CPU-5V in a miniforce chassis, please first disable the VMEbus System Controller function by setting switch SW8-2 to OFF and also setting SW7-4 to OFF.
2.4.2 VMEbus SYSRESET Enable/Disable

2.4.2.1 SYSRESET Input

An external SYSRESET generates an on-board RESET in the default switch setting, i.e., SW7-3 is ON. When SW7-3 is OFF, the external SYSRESET does not generate an on-board RESET.

2.4.2.2 SYSRESET Output

An on-board RESET drives the SYSRESET signal to the VMEbus to low in the default switch setting, i.e., SW7-4 is ON. When SW7-4 is OFF, an on-board RESET doesn’t drive the SYSRESET signal to the VMEbus to low.

CAUTION

Do not switch SW7-4 (SYSRESET output) to ON and SW8-2 (VMEbus Slot-1 device) to OFF at the same time.

The VMEbus Specification requires that if SYSRESET is driven, the SYSRESET signal shall be driven low for at least 200 ms. However, when the CPU-5V is not a VMEbus slot-1 device and the SYSRESET output signal is enabled, then the CPU-5V no longer conforms with this rule.

By default, the SYSRESET output is enabled. In this case it generates the SYSRESET signal to the VMEbus.

2.4.3 Serial Ports

By default, both serial ports are configured as RS-232 interfaces. It is also possible to configure both ports as RS-422 or RS-485 interfaces. This optional configuration is achieved with the special FORCE Hybrids FH-003 and FH-005.

The chapter "Default Switch Settings" on page 12 shows the necessary switch settings for RS-232 operation, where SW4 controls serial port A and SW5 controls serial port B. Please check that the switches are set accordingly.
2.4.4  RESET and ABORT Key Enable

To enable the RESET and the ABORT functions on the front panel, set switches SW7-1 (RESET) and SW7-2 (ABORT) to ON.

2.4.5  SCSI Termination

2.4.5.1  SCSI Termination at the Front Panel

Termination at the front panel for the SCSI interface is automatic when SW6-1 is OFF. This is the default setting. Automatic means that when a SCSI cable is plugged into the front panel connector, the termination is automatically disabled. When there is no SCSI cable plugged into the front panel, then the termination is automatically enabled.

2.4.5.2  SCSI Termination at P2

Termination at the VMEbus P2 for the SCSI interface is enabled when SW6-2 is OFF. This is the default setting.

CAUTION

When installing the SPARC CPU-5V in a MICROFORCE chassis, please first disable the SCSI termination by switching SW6-1 and SW6-2 to ON.
2.4.6 **Boot Flash Memory Write Protection**

Both of the Boot Flash Memory devices are write protectable via the switch SW6-3. When SW6-3 is OFF, the devices are write protected.

2.4.7 **User Flash Memory Write Protection**

The User Flash Memory devices are write protectable via SW6-4. When SW6-4 is OFF, the User Flash Memory devices are write protected.

2.4.8 **Reserved Switches**

SW4-4, SW5-4, and SW8-4 are reserved for test purposes.
2.4.9 Parallel Port or Floppy Interface via VME P2 Connector

Via a 16-pin configuration switch matrix, it is possible for either the parallel port interface or the floppy interface to be available on the VME P2 connector.

The default setting enables the floppy interface via the VME P2 connector, with the configuration switch matrix plugged into B1 and B2. This means, of course, that by default the parallel port interface is not available via the VMEbus P2 connector.

To enable the parallel port interface via the VME P2 connector, plug the configuration switch matrix in sockets B2 and B3.
2.4.10 Ethernet via Front Panel or VME P2 Connector

Via an 8-pin configuration switch matrix, it is either possible for the Ethernet interface to be available via the front panel or the VME P2 connector. The default configuration provides the Ethernet through the front panel connector.

In order to have the Ethernet interface accessible via the VME P2 connector, the default configuration must be changed.

By default, the Ethernet interface is available through the front panel with the configuration switch matrix plugged into connectors B12 and B13.

To configure the Ethernet interface to be accessible from the VMEbus P2 connector, the configuration switch matrix must be plugged into connectors B11 and B12.

FIGURE 5. Ethernet Interface via Front Panel

WARNING: When the Ethernet interface is configured via VMEbus P2, do not connect the Ethernet at the front panel.
2.5 OpenBoot Firmware

This chapter describes the use of OpenBoot firmware. Specifically, you will read how to perform the following tasks.

- Boot the System
- Run Diagnostics
- Display System Information
- Reset the System
- OpenBoot Help

For detailed information concerning OpenBoot, please see the OPEN BOOT PROM 2.0 MANUAL SET. This manual is included in the SPARC CPU-5V Technical Reference Manual Set.

2.5.1 Boot the System

The most important function of OpenBoot firmware is booting the system. Booting is the process of loading and executing a stand-alone program such as the operating system. After it is powered on, the system usually boots automatically after it has passed the Power On SelfTest (POST). This occurs without user intervention.

If necessary, you can explicitly initiate the boot process from the OpenBoot command interpreter. Automatic booting uses the default boot device specified in non-volatile RAM (NVRAM); user initiated booting uses either the default boot device or one specified by the user.

To boot the system from the default boot device, type the following command at the Forth Monitor prompt.

```
ok boot
```

or, if you are at the Restricted Monitor Prompt, you have to type the following:

```
> b
```
The boot command has the following format:

boot [device-specifier] [filename] [-ah]

The optional parameters are described as follows.

[device-specifier] The name (full path or alias) of the boot device. Typical values are cdrom, disk, floppy, net or tape.

[filename] The name of the program to be booted. filename is relative to the root of the selected device. If no filename is specified, the boot command uses the value of boot-file NVRAM parameter. The NVRAM parameters used for booting are described in the following chapter.

[-a] -a prompt interactively for the device and name of the boot file.

[-h] -h halt after loading the program.

NOTE: These options are specific to the operating system and may differ from system to system.

To explicitly boot from the internal disk, type:

ok boot disk

or at the Restricted Monitor prompt:

> b disk
To retrieve a list of all device alias definitions, type `devalias` at the Forth Monitor command prompt. The following table lists some typical device aliases:

**Table 5: Device Alias Definitions**

<table>
<thead>
<tr>
<th>Alias</th>
<th>Boot Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk</td>
<td>/iommu/sbus/espdma/esp/sd@3,0</td>
<td>Default disk (1st internal) SCSI-ID 3</td>
</tr>
<tr>
<td>disk3</td>
<td>/iommu/sbus/espdma/esp/sd@3,0</td>
<td>First internal disk SCSI-ID 3</td>
</tr>
<tr>
<td>disk2</td>
<td>/iommu/sbus/espdma/esp/sd@2,0</td>
<td>Additional internal disk SCSI-ID 2</td>
</tr>
<tr>
<td>disk1</td>
<td>/iommu/sbus/espdma/esp/sd@1,0</td>
<td>External disk SCSI-ID 1</td>
</tr>
<tr>
<td>disk0</td>
<td>/iommu/sbus/espdma/esp/sd@0,0</td>
<td>External disk SCSI-ID 0</td>
</tr>
<tr>
<td>tape</td>
<td>/iommu/sbus/espdma/esp/st@4,0</td>
<td>First tape drive SCSI-ID 4</td>
</tr>
<tr>
<td>tape0</td>
<td>/iommu/sbus/espdma/esp/st@4,0</td>
<td>First tape drive SCSI-ID 4</td>
</tr>
<tr>
<td>tape1</td>
<td>/iommu/sbus/espdma/esp/st@5,0</td>
<td>Second tape drive SCSI-ID 5</td>
</tr>
<tr>
<td>cdrom</td>
<td>/iommu/sbus/espdma/esp/sd@6,0:d</td>
<td>CD-ROM partition d, SCSI-ID 6</td>
</tr>
<tr>
<td>net</td>
<td>/iommu/sbus/ledma/le</td>
<td>Ethernet</td>
</tr>
<tr>
<td>floppy</td>
<td>/obio/SUNW,fdtwo</td>
<td>Floppy drive</td>
</tr>
</tbody>
</table>
2.5.2 NVRAM Boot Parameters

The OpenBoot firmware holds configuration parameters in NVRAM. At the Forth Monitor prompt, type `printenv` to see a list of all available configuration parameters. The OpenBoot command `setenv` may be used to set these parameters.

```
setenv [configuration parameter] [value]
```

This information refers only to those configuration parameters which are involved in the boot process. The following table lists these parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto-boot?</td>
<td>true</td>
<td>If true, boot automatically after power on or reset</td>
</tr>
<tr>
<td>boot-device</td>
<td>disk</td>
<td>Device from which to boot</td>
</tr>
<tr>
<td>boot-file</td>
<td>empty string</td>
<td>File to boot</td>
</tr>
<tr>
<td>diag-switch?</td>
<td>false</td>
<td>If true, run in diagnostic mode</td>
</tr>
<tr>
<td>diag-device</td>
<td>net</td>
<td>Device from which to boot in diagnostic mode</td>
</tr>
<tr>
<td>diag-file</td>
<td>empty string</td>
<td>File to boot in diagnostic mode</td>
</tr>
</tbody>
</table>

When booting an operating system or another stand-alone program, and neither a boot device nor a filename is supplied, the boot command of the Forth Monitor takes the omitted values from the NVRAM configuration parameters. If the parameter `diag-switch?` is false, `boot-device` and `boot-file` are used. Otherwise, the OpenBoot firmware uses `diag-device` and `diag-file` for booting.

For a detailed description of all NVRAM configuration parameters, please refer to the *OPEN BOOT PROM 2.0 MANUAL*.
2.5.3 Diagnostics

At power on or after reset, the OpenBoot firmware executes POST. If the NVRAM configuration parameter diag-switch? is true for each test, a message is displayed on a terminal connected to the first serial port. In case the system is not working correctly, error messages indicating the problem are displayed. After POST, the OpenBoot firmware boots an operating system or enters the Forth Monitor if the NVRAM configuration parameter auto-boot? is false.

The Forth Monitor includes several diagnostic routines. These on-board tests let you check devices such as network controller, SCSI devices, floppy disk system, memory, clock and installed SBus cards. User installed devices can be tested if their firmware includes a selftest routine.

The table below lists several diagnostic routines.
Table 7: Diagnostic Routines

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>probe-scsi</td>
<td>Identify devices connected to the on-board SCSI bus</td>
</tr>
<tr>
<td>probe-scsi-all [device-path]</td>
<td>Perform probe-scsi on all SCSI buses installed in the system below the specified device tree node. (If device-path is omitted, the root node is used.)</td>
</tr>
<tr>
<td>test  device-specifier</td>
<td>Execute the specified device’s selftest method. device-specifier may be a device path name or a device alias. For example: test net - test network connection test /memory - test number of megabytes specified in the selftest-#megs NVRAM parameter or test all of memory if diag-switch? is true</td>
</tr>
<tr>
<td>test-all [device-specifier]</td>
<td>Test all devices (that have a built-in selftest method) below the specified device tree node. (If device-path is omitted, the root node is used.)</td>
</tr>
<tr>
<td>watch-clock</td>
<td>Monitor the clock function</td>
</tr>
<tr>
<td>watch-net</td>
<td>Monitor network connection</td>
</tr>
</tbody>
</table>

To check the on-board SCSI bus for connected devices, type:

```
ok probe-scsi
Target 3
    Unit 0 Disk MICROP 1684-07MB1036511ASOC1684
ok
```

To test all the SCSI buses installed in the system, type:

```
ok probe-scsi-all
/iommu@0,10000000/sbus@0,10001000/esp@2,100000
Target 6
    Unit 0 Disk Removable Read Only Device SONY CD-ROM CDU-8012 3.1a
/iommu@0,10000000/sbus@0,10001000/espdma@4,84000000/esp@4,88000000
Target 3
    Unit 0 Disk MICROP 1684-07MB1036511ASOC1684
ok
```

The actual response depends on the devices on the SCSI buses.
To test a single installed device, type:

```
ok test device-specifier
```

This executes the device method name selftest of the specified device node. device-specifier may be a device path name or a device alias as described in Table 5, “Device Alias Definitions,” on page 22. The response depends on the selftest of the device node.

To test a group of installed devices, type:

```
ok test-all
```

All devices below the root node of the device tree are tested. The response depends on the devices that have a selftest routine. If a device specifier option is supplied at the command line, all devices below the specified device tree node are tested.

When you use the memory testing routine, the system tests the number of megabytes of memory specified in the NVRAM configuration parameter selftest-#megs. If the NVRAM configuration parameter diag-switch? is true, all memory is tested.

```
ok test /memory
  testing 32 meg of memory at addr 0 27
  ok
```

The command test-memory is equivalent to test /memory. In the example above, the first number (0) is the base address of the memory bank to be tested, the second number (27) is the number of megabytes remaining. If the CPU board is working correctly, the memory is erased and tested and you will receive the ok prompt. If the PROM or the on-board memory is not working, you receive one of a number of possible error messages indicating the problem.

To test the clock function, type:

```
ok watch-clock
  Watching the ‘seconds’ register of the real time clock chip.
  It should be ‘ticking’ once a second.
  Type any key to stop.
  22
  ok
```

The system responds by incrementing a number once a second. Press any key to stop the test.
To monitor the network connection, type:

```
ok watch-net
Using AUI Ethernet Interface
Lance register test -- succeeded.
Internal loopback test -- succeeded.
External loopback test -- succeeded.
Looking for Ethernet packets.
'.' is a good packet. 'X' is a bad packet.
Type any key to stop.
........X............................X.........
ok
```

The system monitors the network traffic, displaying "." each time it receives a valid packet and displaying "X" each time it receives a packet with an error that can be detected by the network hardware interface.
2.5.4 Display System Information

The Forth Monitor provides several commands to display system information. These commands let you display the system banner, the Ethernet address for the Ethernet controller, the contents of the ID PROM, and the version number of the OpenBoot firmware.

The ID PROM contains information specific to each individual machine, including the serial number, date of manufacture, and assigned Ethernet address.

The following table lists these commands.
Table 8: Commands to Display System Information

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>banner</td>
<td>Display system banner.</td>
</tr>
<tr>
<td>show-sbus</td>
<td>Display list of installed and probed SBus devices.</td>
</tr>
<tr>
<td>.enet-addr</td>
<td>Display current Ethernet address.</td>
</tr>
<tr>
<td>.idprom</td>
<td>Display ID PROM contents, formatted.</td>
</tr>
<tr>
<td>.traps</td>
<td>Display a list of SPARC trap types.</td>
</tr>
<tr>
<td>.version</td>
<td>Display version and date of the Boot PROM.</td>
</tr>
<tr>
<td>show-devs</td>
<td>Display a list of all device tree nodes.</td>
</tr>
<tr>
<td>devalias</td>
<td>Display a list of all device aliases.</td>
</tr>
</tbody>
</table>
2.5.5  Reset the System

If your system needs to be reset, you either press the reset button on the front panel or, if you are in the Forth Monitor, type reset on the command line.

```
ok reset
```

The system immediately begins executing the Power On SelfTest (POST) and initialization procedures. Once the POST finishes, the system either boots automatically or enters the Forth Monitor, just as it would have done after a power-on cycle.

2.5.6  OpenBoot Help

The Forth Monitor contains an on-line help. To get this, type:

```
ok help
Enter 'help command-name' or 'help category-name' for more help
(Use ONLY the first word of a category description)
Examples: help select -or- help line
Main categories are:
File download and boot
Resume execution
Diag (diagnostic routines)
Power on reset
> - prompt
Floppy eject
Select I/O devices
Ethernet
System and boot configuration parameters
Line editor
Tools (memory, numbers, new commands, loops)
Assembly debugging (breakpoints, registers, disassembly, symbolic)
Sync (synchronize disk data)
Nvramrc (making new commands permanent)
ok
```

A list of all available help categories is displayed. These categories may also contain subcategories. To get help for special forth words or subcategories just type help [name]. An example is shown on the next page.
A example of how to get help for special forth words or subcategories:

```
ok help tools
Category: Tools (memory,numbers,new commands,loops)
Sub-categories are:
Memory access
Arithmetic
Radix (number base conversions)
Numeric output
Defining new commands
Repeated loops
ok
ok help memory
Category: Memory access
dump ( addr length -- ) display memory at addr for length bytes
fill ( addr length byte -- ) fill memory starting at addr with byte
move ( src dest length -- ) copy length bytes from src to dest address
map? ( vaddr -- ) show memory map information for the virtual address
I? ( addr - ) display the 32-bit number from location addr
w? ( addr -- ) display the 16-bit number from location addr
c? ( addr -- ) display the 8-bit number from location addr
l@ (addr - n ) place on the stack the 32-bit data at location addr
w@ ( addr - n ) place on the stack the 16-bit data at location addr
c@ (addr - n ) place on the stack the 8-bit data at location addr
l! ( n addr -- ) store the 32-bit value n at location addr
w! ( n addr - ) store the 16-bit value n at location addr
c! (n addr - ) store the 8-bit value n at location addr
ok
```

The on-line help shows you the forth word, the parameter stack before and after execution of the forth word ( before -- after), and a short description.

The on-line help of the Forth Monitor is located in the boot PROM, so there is not an online help for all forth words.
2.6 Front Panel

FIGURE 6. Diagram of the Front Panel
2.6.1 Features of the Front Panel

The features listed below are described in detail in Section 3 of the *SPARC CPU-5V Technical Reference Manual*.

Table 9: Front Panel Layout

<table>
<thead>
<tr>
<th>Device</th>
<th>Function</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>Reset</td>
<td>RESET</td>
</tr>
<tr>
<td>Switch</td>
<td>Abort</td>
<td>ABORT</td>
</tr>
<tr>
<td>HEX. Display</td>
<td>Diagnostic</td>
<td>DIAG</td>
</tr>
<tr>
<td>Rotary Switch</td>
<td>User defined</td>
<td>MODE</td>
</tr>
<tr>
<td>LED/LED</td>
<td>RUN/HALT VME Busmaster/SYSFAIL</td>
<td>RUN BM</td>
</tr>
<tr>
<td>LED/LED</td>
<td>Software programmable</td>
<td>01</td>
</tr>
<tr>
<td>Mini DIN Connector</td>
<td>Keyboard/Mouse</td>
<td>KBD</td>
</tr>
<tr>
<td>Serial Connector</td>
<td>Serial Interfaces</td>
<td>SERIAL A+B</td>
</tr>
<tr>
<td>SCSI Connector</td>
<td>SCSI Interface</td>
<td>SCSI</td>
</tr>
<tr>
<td>D-Sub Connector</td>
<td>Ethernet</td>
<td>ETHERNET</td>
</tr>
</tbody>
</table>
2.7 SPARC CPU-5V Connectors

The connectors on the SPARC CPU-5V are listed in the following table.

Table 10: SPARC CPU-5V Connectors

<table>
<thead>
<tr>
<th>Function</th>
<th>Location</th>
<th>Type</th>
<th>Manufacturer Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>Front Panel</td>
<td>15-pin D-Sub</td>
<td>AMP 747845-4</td>
</tr>
<tr>
<td>Serial Port A + B</td>
<td>Front Panel</td>
<td>26-pin Fine Pitch</td>
<td>AMP 749831-2</td>
</tr>
<tr>
<td>SCSI</td>
<td>Front Panel</td>
<td>50-pin Fine Pitch</td>
<td>AMP 749831-5</td>
</tr>
<tr>
<td>Keyboard/Mouse</td>
<td>Front Panel</td>
<td>8-pin Mini DIN</td>
<td>AMP 749232-1</td>
</tr>
<tr>
<td>SBus Slot2 (SBus Slave Select 1)</td>
<td>P3</td>
<td>96-pin SMD</td>
<td>FUJITSU FCN-234J096-G/V</td>
</tr>
<tr>
<td>SBus Slot3 (SBus Slave Select 2)</td>
<td>P4</td>
<td>96-pin SMD</td>
<td>FUJITSU FCN-234J096-G/V</td>
</tr>
<tr>
<td>VMEbus P1</td>
<td>P1</td>
<td>96-pin VGA</td>
<td>Various</td>
</tr>
<tr>
<td>VMEbus P2</td>
<td>P2</td>
<td>96-pin VGA</td>
<td>Various</td>
</tr>
</tbody>
</table>

The following pages show the pinouts of the connectors.
2.7.1 Ethernet Connector Pinout

The following table is a pinout of the Ethernet connector. The figure below shows the Ethernet connector and pin numbers.

Table 11: Ethernet Connector Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Analog GND</td>
</tr>
<tr>
<td>2</td>
<td>Collision+</td>
</tr>
<tr>
<td>3</td>
<td>Transmit Data+</td>
</tr>
<tr>
<td>4</td>
<td>Analog GND</td>
</tr>
<tr>
<td>5</td>
<td>Receive Data+</td>
</tr>
<tr>
<td>6</td>
<td>Analog GND</td>
</tr>
<tr>
<td>7</td>
<td>N.C.</td>
</tr>
<tr>
<td>8</td>
<td>Analog GND</td>
</tr>
<tr>
<td>9</td>
<td>Collision-</td>
</tr>
<tr>
<td>10</td>
<td>Transmit Data-</td>
</tr>
<tr>
<td>11</td>
<td>Analog GND</td>
</tr>
<tr>
<td>12</td>
<td>Receive Data-</td>
</tr>
<tr>
<td>13</td>
<td>+12VDC</td>
</tr>
<tr>
<td>14</td>
<td>Analog GND</td>
</tr>
<tr>
<td>15</td>
<td>N.C.</td>
</tr>
</tbody>
</table>

FIGURE 7. Pinout of the Ethernet Cable Connector
### 2.7.2 Serial Port A and B Connector Pinout

The following table is a pinout of the serial port connector. The figure on the next page shows the serial port connector and location of the pin numbers.

**Table 12: Serial Port A and B Connector Pinout**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>none</td>
<td>none</td>
<td>A</td>
<td>Not connected</td>
</tr>
<tr>
<td>2</td>
<td>TD</td>
<td>output</td>
<td>A</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>3</td>
<td>RD</td>
<td>input</td>
<td>A</td>
<td>Receive Data</td>
</tr>
<tr>
<td>4</td>
<td>RTS</td>
<td>output</td>
<td>A</td>
<td>Request To Send</td>
</tr>
<tr>
<td>5</td>
<td>CTS</td>
<td>input</td>
<td>A</td>
<td>Clear To Send</td>
</tr>
<tr>
<td>6</td>
<td>DSR</td>
<td>input</td>
<td>A</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>7</td>
<td>SG</td>
<td>none</td>
<td>A</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>DCD</td>
<td>input</td>
<td>A</td>
<td>Data Carrier Detect</td>
</tr>
<tr>
<td>9</td>
<td>none</td>
<td>none</td>
<td>Not connected</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>none</td>
<td>none</td>
<td>Not connected</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SDTR</td>
<td>output</td>
<td>B</td>
<td>Secondary Data Terminal Ready</td>
</tr>
<tr>
<td>12</td>
<td>SDCD</td>
<td>input</td>
<td>B</td>
<td>Secondary Data Carrier Detect</td>
</tr>
<tr>
<td>13</td>
<td>SCTS</td>
<td>input</td>
<td>B</td>
<td>Secondary Clear To Send</td>
</tr>
<tr>
<td>14</td>
<td>STD</td>
<td>output</td>
<td>B</td>
<td>Secondary Transmit Data</td>
</tr>
<tr>
<td>15</td>
<td>TC</td>
<td>input</td>
<td>A</td>
<td>Transmit Clock: DCE source</td>
</tr>
<tr>
<td>16</td>
<td>SRD</td>
<td>input</td>
<td>B</td>
<td>Secondary Receive Data</td>
</tr>
<tr>
<td>17</td>
<td>RC</td>
<td>input</td>
<td>A</td>
<td>Receive Clock</td>
</tr>
<tr>
<td>18</td>
<td>STC</td>
<td>input</td>
<td>B</td>
<td>Secondary Transmit Clock</td>
</tr>
<tr>
<td>19</td>
<td>SKTS</td>
<td>output</td>
<td>B</td>
<td>Secondary Request To Send</td>
</tr>
<tr>
<td>20</td>
<td>DTR</td>
<td>output</td>
<td>A</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>21</td>
<td>SDSR</td>
<td>input</td>
<td>B</td>
<td>Secondary Data Terminal Ready*</td>
</tr>
<tr>
<td>22</td>
<td>SRC</td>
<td>input</td>
<td>B</td>
<td>Secondary Receive Clock*</td>
</tr>
<tr>
<td>23</td>
<td>SSG</td>
<td>none</td>
<td>B</td>
<td>Secondary Signal Ground</td>
</tr>
<tr>
<td>24</td>
<td>TC</td>
<td>output</td>
<td>A</td>
<td>Transmit Clock: DTE source</td>
</tr>
<tr>
<td>25</td>
<td>STC</td>
<td>output</td>
<td>B</td>
<td>Transmit Clock: DTE source</td>
</tr>
</tbody>
</table>
FIGURE 8. Serial Ports A and B Connector Pinout
2.7.3 **SCSI Connector Pinout**

The following table is a pinout of the SCSI connector. The figure on the next page shows the SCSI connector and location of the pin numbers.

### Table 13: SCSI 50-Pin Connector

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Signal</th>
<th>Pin No.</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>26</td>
<td>SCSI Data 0</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>27</td>
<td>SCSI Data 1</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>28</td>
<td>SCSI Data 2</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>29</td>
<td>SCSI Data 3</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>30</td>
<td>SCSI Data 4</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
<td>31</td>
<td>SCSI Data 5</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>32</td>
<td>SCSI Data 6</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>33</td>
<td>SCSI Data 7</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td>34</td>
<td>SCSI DP</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>35</td>
<td>GND</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
<td>36</td>
<td>DISABLE TERM</td>
</tr>
<tr>
<td>12</td>
<td>N.C.</td>
<td>37</td>
<td>N.C.</td>
</tr>
<tr>
<td>13</td>
<td>N.C.</td>
<td>38</td>
<td>TERM_PWR</td>
</tr>
<tr>
<td>14</td>
<td>N.C.</td>
<td>39</td>
<td>N.C.</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>40</td>
<td>GND</td>
</tr>
<tr>
<td>16</td>
<td>GND</td>
<td>41</td>
<td>SCSI ATN</td>
</tr>
<tr>
<td>17</td>
<td>GND</td>
<td>42</td>
<td>GND</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>43</td>
<td>SCSI BSY</td>
</tr>
<tr>
<td>19</td>
<td>GND</td>
<td>44</td>
<td>SCSI ACK</td>
</tr>
<tr>
<td>20</td>
<td>GND</td>
<td>45</td>
<td>SCSI RST</td>
</tr>
<tr>
<td>21</td>
<td>GND</td>
<td>46</td>
<td>SCSI MSG</td>
</tr>
<tr>
<td>22</td>
<td>GND</td>
<td>47</td>
<td>SCSI SEL</td>
</tr>
<tr>
<td>23</td>
<td>GND</td>
<td>48</td>
<td>SCSI CD</td>
</tr>
<tr>
<td>24</td>
<td>GND</td>
<td>49</td>
<td>SCSI REQ</td>
</tr>
<tr>
<td>25</td>
<td>GND</td>
<td>50</td>
<td>SCSI IO</td>
</tr>
</tbody>
</table>
FIGURE 9. Pinout of SCSI Connector

![Pinout Diagram]

- Pins 25 and 50
- Pin 1 is highlighted
- Pin 26 is highlighted
2.7.4 Keyboard/Mouse Connector Pinout

The following table is a pinout of the keyboard/mouse connector. The keyboard and mouse port is available on the front panel via a Mini DIN connector.

Table 14: Keyboard/Mouse Connector Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>+SVDC</td>
</tr>
<tr>
<td>4</td>
<td>Mouse In</td>
</tr>
<tr>
<td>5</td>
<td>Keyboard Out</td>
</tr>
<tr>
<td>6</td>
<td>Keyboard In</td>
</tr>
<tr>
<td>7</td>
<td>Mouse Out</td>
</tr>
<tr>
<td>8</td>
<td>+SVDC</td>
</tr>
</tbody>
</table>

FIGURE 10. Keyboard/Mouse Connector
### 2.7.5 VME P2 Connector Pinout

The following table is a pinout of the VME P2 connector.

#### Table 15: VME P2 Connector Pinout

<table>
<thead>
<tr>
<th>PIN</th>
<th>Row A</th>
<th>Row C (Floppy drive available via switch matrix connected to sockets B1 &amp; B2)</th>
<th>Row C (Parallel Port available via switch matrix connected to sockets B2 &amp; B3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCSI Data 0</td>
<td>FLPY DENSEL</td>
<td>CENTR D5</td>
</tr>
<tr>
<td>2</td>
<td>SCSI Data 1</td>
<td>FLPY DENSENSE</td>
<td>CENTR D0</td>
</tr>
<tr>
<td>3</td>
<td>SCSI Data 2</td>
<td>CENTR D1</td>
<td>CENTR D1</td>
</tr>
<tr>
<td>4</td>
<td>SCSI Data 3</td>
<td>FLPY INDEX</td>
<td>CENTR D2</td>
</tr>
<tr>
<td>5</td>
<td>SCSI Data 4</td>
<td>FLPY DRVSEL</td>
<td>CENTR D3</td>
</tr>
<tr>
<td>6</td>
<td>SCSI Data 5</td>
<td>CENTR D4</td>
<td>CENTR D4</td>
</tr>
<tr>
<td>7</td>
<td>SCSI Data 6</td>
<td>CENTR D5</td>
<td>CENTR D5</td>
</tr>
<tr>
<td>8</td>
<td>SCSI Data 7</td>
<td>FLPY MOTEN</td>
<td>CENTR D6</td>
</tr>
<tr>
<td>9</td>
<td>SCSI DP</td>
<td>FLPY DIR</td>
<td>CENTR D7</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>FLPY STEP</td>
<td>CENTR ACK</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
<td>FLPY WRDATA</td>
<td>CENTR BSY</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
<td>FLPY WRGATE</td>
<td>CENTR FE</td>
</tr>
<tr>
<td>13</td>
<td>TERMPEWR</td>
<td>FLPY TRCK0</td>
<td>CENTR AF</td>
</tr>
<tr>
<td>14</td>
<td>GND</td>
<td>FLPY WRPROT</td>
<td>CENTR INIT</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>FLPY RDDATA</td>
<td>CENTR ERR</td>
</tr>
<tr>
<td>16</td>
<td>SCSI ATN</td>
<td>FLPY HEADSEL</td>
<td>CENT SLCT IN</td>
</tr>
<tr>
<td>17</td>
<td>GND</td>
<td>FLPY DISKCHG</td>
<td>CENT SLCT</td>
</tr>
<tr>
<td>18</td>
<td>SCSI BSY</td>
<td>FLPY EJECT</td>
<td>N.C.</td>
</tr>
<tr>
<td>19</td>
<td>SCSI ACK</td>
<td>ETH +12V</td>
<td>ETH +12V</td>
</tr>
<tr>
<td>20</td>
<td>SCSI RST</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>SCSI MSG</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>22</td>
<td>SCSI SEL</td>
<td>ETH REC+2</td>
<td>ETH REC+2</td>
</tr>
<tr>
<td>23</td>
<td>SCSI CD</td>
<td>ETH REC-2</td>
<td>ETH REC-2</td>
</tr>
<tr>
<td>24</td>
<td>SCSI REQ</td>
<td>ETH TRA+2</td>
<td>ETH TRA+2</td>
</tr>
<tr>
<td>25</td>
<td>SCSI IO</td>
<td>ETH TRA-2</td>
<td>ETH TRA-2</td>
</tr>
<tr>
<td>26</td>
<td>Mouse IN</td>
<td>ETH COL+2</td>
<td>ETH COL+2</td>
</tr>
<tr>
<td>27</td>
<td>Keyboard OUT</td>
<td>ETH COL-2</td>
<td>ETH COL-2</td>
</tr>
<tr>
<td>28</td>
<td>Keyboard IN</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>29</td>
<td>TXD Port A</td>
<td>TXD Port B</td>
<td>TXD Port B</td>
</tr>
<tr>
<td>30</td>
<td>RXD Port A</td>
<td>RXD Port B</td>
<td>RXD Port B</td>
</tr>
<tr>
<td>31</td>
<td>RTS Port A</td>
<td>RTS Port B</td>
<td>RTS Port B</td>
</tr>
<tr>
<td>32</td>
<td>CTS Port A</td>
<td>CTS Port B</td>
<td>CTS Port B</td>
</tr>
</tbody>
</table>
2.7.6 The IOBP-10 Connectors

The IOBP-10 is an I/O back panel on VMEbus P2 with flat cable connectors for SCSI, serial I/O, Centronics/floppy interface, and a micro D-Sub connector for an Ethernet interface. This back panel can be plugged into the VMEbus P2 connector. The diagram below shows all the connectors. This IOBP-10 back panel is especially designed for the SPARC CPU-5V. Do not use any other I/O back panels on the SPARC CPU-5V, for example, the IOBP-1.

FIGURE 11. The IOBP-10

The pinouts of the connectors (P1) ... (P6) are shown in the following tables.

CAUTION

This IOBP-10 back panel is especially designed for the SPARC CPU-5V. Do not use any other I/O back panels on the SPARC CPU-5V, for example, the IOBP-1.
### Table 16: IOBP-10 P1 Pinout

<table>
<thead>
<tr>
<th>ROW A</th>
<th>Signal</th>
<th>ROW B</th>
<th>Signal</th>
<th>ROW C</th>
<th>Signal for Floppy Interface</th>
<th>Signal for Parallel Port Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCSI Data 0</td>
<td>1</td>
<td>N.C.</td>
<td>1</td>
<td>FPY DENSEL</td>
<td>CENTR DS</td>
</tr>
<tr>
<td>2</td>
<td>SCSI Data 1</td>
<td>2</td>
<td>GND</td>
<td>2</td>
<td>FPY DENSENS</td>
<td>CENTR Data 0</td>
</tr>
<tr>
<td>3</td>
<td>SCSI Data 2</td>
<td>3</td>
<td>N.C.</td>
<td>3</td>
<td>N.C.</td>
<td>CENTR Data 1</td>
</tr>
<tr>
<td>4</td>
<td>SCSI Data 3</td>
<td>4</td>
<td>N.C.</td>
<td>4</td>
<td>FPY INDEX</td>
<td>CENTR Data 2</td>
</tr>
<tr>
<td>5</td>
<td>SCSI Data 4</td>
<td>5</td>
<td>N.C.</td>
<td>5</td>
<td>FPY DRVSEL</td>
<td>CENTR Data 3</td>
</tr>
<tr>
<td>6</td>
<td>SCSI Data 5</td>
<td>6</td>
<td>N.C.</td>
<td>6</td>
<td>N.C.</td>
<td>CENTR Data 4</td>
</tr>
<tr>
<td>7</td>
<td>SCSI Data 6</td>
<td>7</td>
<td>N.C.</td>
<td>7</td>
<td>N.C.</td>
<td>CENTR Data 5</td>
</tr>
<tr>
<td>8</td>
<td>SCSI Data 7</td>
<td>8</td>
<td>N.C.</td>
<td>8</td>
<td>FPY MOTEN</td>
<td>CENTR Data 6</td>
</tr>
<tr>
<td>9</td>
<td>SCSI DP</td>
<td>9</td>
<td>N.C.</td>
<td>9</td>
<td>FPY DIR</td>
<td>CENTR Data 7</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>10</td>
<td>N.C.</td>
<td>10</td>
<td>FPY STEP</td>
<td>CENTR ACK</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
<td>11</td>
<td>N.C.</td>
<td>11</td>
<td>FPY WRDATA</td>
<td>CENTR BSY</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
<td>12</td>
<td>GND</td>
<td>12</td>
<td>FPY WRGATE</td>
<td>CENTR PE</td>
</tr>
<tr>
<td>13</td>
<td>TERMPWR</td>
<td>13</td>
<td>N.C.</td>
<td>13</td>
<td>FPY TRACK0</td>
<td>CENTR AF</td>
</tr>
<tr>
<td>14</td>
<td>GND</td>
<td>14</td>
<td>N.C.</td>
<td>14</td>
<td>FPY WRPROT</td>
<td>CENTR INIT</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>15</td>
<td>N.C.</td>
<td>15</td>
<td>FPY RDDATA</td>
<td>CENTR ERR</td>
</tr>
<tr>
<td>16</td>
<td>SCSI ATN</td>
<td>16</td>
<td>N.C.</td>
<td>16</td>
<td>FPY HEADSEL</td>
<td>CENTR SLCT IN</td>
</tr>
<tr>
<td>17</td>
<td>GND</td>
<td>17</td>
<td>N.C.</td>
<td>17</td>
<td>FPY DISKCHG</td>
<td>CENTR SLCT</td>
</tr>
<tr>
<td>18</td>
<td>SCSI BSY</td>
<td>18</td>
<td>N.C.</td>
<td>18</td>
<td>FPY EJECT</td>
<td>RESERVED</td>
</tr>
<tr>
<td>19</td>
<td>SCSI ACK</td>
<td>19</td>
<td>N.C.</td>
<td>19</td>
<td>+12VDC²</td>
<td>+12VDC²</td>
</tr>
<tr>
<td>20</td>
<td>SCSI RST</td>
<td>20</td>
<td>N.C.</td>
<td>20</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>SCSI MSG</td>
<td>21</td>
<td>N.C.</td>
<td>21</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>22</td>
<td>SCSI SEL</td>
<td>22</td>
<td>GND</td>
<td>22</td>
<td>ETH REC+²</td>
<td>ETH REC+²</td>
</tr>
<tr>
<td>23</td>
<td>SCSI CD</td>
<td>23</td>
<td>N.C.</td>
<td>23</td>
<td>ETH REC-²</td>
<td>ETH REC-²</td>
</tr>
<tr>
<td>24</td>
<td>SCSI REQ</td>
<td>24</td>
<td>N.C.</td>
<td>24</td>
<td>ETH TRA+²</td>
<td>ETH TRA+²</td>
</tr>
<tr>
<td>25</td>
<td>SCSI IO</td>
<td>25</td>
<td>N.C.</td>
<td>25</td>
<td>ETH TRA-²</td>
<td>ETH TRA-²</td>
</tr>
<tr>
<td>26</td>
<td>Mouse IN</td>
<td>26</td>
<td>N.C.</td>
<td>26</td>
<td>ETH COL+²</td>
<td>ETH COL+²</td>
</tr>
</tbody>
</table>
### Table 16: IOBP-10 P1 Pinout (Continued)

<table>
<thead>
<tr>
<th>ROW A</th>
<th>Signal</th>
<th>ROW B</th>
<th>Signal</th>
<th>ROW C</th>
<th>Signal for Floppy Interface</th>
<th>Signal for Parallel Port Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Keyboard Out</td>
<td>27</td>
<td>N.C.</td>
<td>27</td>
<td>ETH COL- ²</td>
<td>ETH COL- ²</td>
</tr>
<tr>
<td>28</td>
<td>Keyboard In</td>
<td>28</td>
<td>N.C.</td>
<td>28</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>29</td>
<td>TxD Port A</td>
<td>29</td>
<td>N.C.</td>
<td>29</td>
<td>TxD Port B</td>
<td>TxD Port B</td>
</tr>
<tr>
<td>30</td>
<td>RxD Port A</td>
<td>30</td>
<td>N.C.</td>
<td>30</td>
<td>RxD Port B</td>
<td>RxD Port B</td>
</tr>
<tr>
<td>31</td>
<td>RTS Port A</td>
<td>31</td>
<td>GND</td>
<td>31</td>
<td>RTS Port B</td>
<td>RTS Port B</td>
</tr>
<tr>
<td>32</td>
<td>CTS Port A</td>
<td>32</td>
<td>N.C.</td>
<td>32</td>
<td>CTS Port B</td>
<td>CTS Port B</td>
</tr>
</tbody>
</table>

1) For further information, see “Floppy Interface Via VME P2 Connector” on page 18
2) For further information, please see “Ethernet Interface via Front Panel” on page 19
Table 17: IOBP-10 P2 Pinout (SCSI)

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Signal</th>
<th>Pin No.</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>2</td>
<td>SCSI Data 0</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>4</td>
<td>SCSI Data 1</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>6</td>
<td>SCSI Data 2</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>8</td>
<td>SCSI Data 3</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td>10</td>
<td>SCSI Data 4</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
<td>12</td>
<td>SCSI Data 5</td>
</tr>
<tr>
<td>13</td>
<td>GND</td>
<td>14</td>
<td>SCSI Data 6</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>16</td>
<td>SCSI Data 7</td>
</tr>
<tr>
<td>17</td>
<td>GND</td>
<td>18</td>
<td>SCSI DP</td>
</tr>
<tr>
<td>19</td>
<td>GND</td>
<td>20</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>GND</td>
<td>22</td>
<td>GND</td>
</tr>
<tr>
<td>23</td>
<td>GND</td>
<td>24</td>
<td>GND</td>
</tr>
<tr>
<td>25</td>
<td>GND</td>
<td>26</td>
<td>TERMPWR</td>
</tr>
<tr>
<td>27</td>
<td>N.C.</td>
<td>28</td>
<td>GND</td>
</tr>
<tr>
<td>29</td>
<td>GND</td>
<td>30</td>
<td>GND</td>
</tr>
<tr>
<td>31</td>
<td>GND</td>
<td>32</td>
<td>SCSI ATN</td>
</tr>
<tr>
<td>33</td>
<td>GND</td>
<td>34</td>
<td>GND</td>
</tr>
<tr>
<td>35</td>
<td>GND</td>
<td>36</td>
<td>SCSI BSY</td>
</tr>
<tr>
<td>37</td>
<td>GND</td>
<td>38</td>
<td>SCSI ACK</td>
</tr>
<tr>
<td>39</td>
<td>GND</td>
<td>40</td>
<td>SCSI RST</td>
</tr>
<tr>
<td>41</td>
<td>GND</td>
<td>42</td>
<td>SCSI MSG</td>
</tr>
<tr>
<td>43</td>
<td>GND</td>
<td>44</td>
<td>SCSI SEL</td>
</tr>
<tr>
<td>45</td>
<td>GND</td>
<td>46</td>
<td>SCSI CD</td>
</tr>
<tr>
<td>47</td>
<td>GND</td>
<td>48</td>
<td>SCSI REQ</td>
</tr>
<tr>
<td>49</td>
<td>GND</td>
<td>50</td>
<td>SCSI IO</td>
</tr>
</tbody>
</table>
Table 18: IOBP-10 P3 Pinout (Floppy)

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Signal</th>
<th>Pin No.</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FPY EJECT</td>
<td>2</td>
<td>FPY DENSEL</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>4</td>
<td>FPY DENSENS</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>6</td>
<td>N.C.</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>8</td>
<td>FPY INDEX</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td>10</td>
<td>FPY DRVSEL</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
<td>12</td>
<td>N.C.</td>
</tr>
<tr>
<td>13</td>
<td>GND</td>
<td>14</td>
<td>N.C.</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>16</td>
<td>FPY MOTEN</td>
</tr>
<tr>
<td>17</td>
<td>GND</td>
<td>18</td>
<td>FPY DIR</td>
</tr>
<tr>
<td>19</td>
<td>GND</td>
<td>20</td>
<td>FPY STEP</td>
</tr>
<tr>
<td>21</td>
<td>GND</td>
<td>22</td>
<td>FPY WRDATA</td>
</tr>
<tr>
<td>23</td>
<td>GND</td>
<td>24</td>
<td>FPY WRGATE</td>
</tr>
<tr>
<td>25</td>
<td>GND</td>
<td>26</td>
<td>FPY TRACK0</td>
</tr>
<tr>
<td>27</td>
<td>N.C.</td>
<td>28</td>
<td>FPY WRPROT</td>
</tr>
<tr>
<td>29</td>
<td>GND</td>
<td>30</td>
<td>FPY RDDATA</td>
</tr>
<tr>
<td>31</td>
<td>GND</td>
<td>32</td>
<td>FPY HEADSEL</td>
</tr>
<tr>
<td>33</td>
<td>GND</td>
<td>34</td>
<td>FPY DISKCHG</td>
</tr>
</tbody>
</table>

Page 46 FORCCE COMPUTERS
Table 19: IOBP-10 P4 Pinout (Centronics)

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Signal</th>
<th>Pin No.</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CENTR DS</td>
<td>2</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>CENTR Data 0</td>
<td>4</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>CENTR Data 1</td>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>7</td>
<td>CENTR Data 2</td>
<td>8</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>CENTR Data 3</td>
<td>10</td>
<td>GND</td>
</tr>
<tr>
<td>11</td>
<td>CENTR Data 4</td>
<td>12</td>
<td>GND</td>
</tr>
<tr>
<td>13</td>
<td>CENTR Data 5</td>
<td>14</td>
<td>GND</td>
</tr>
<tr>
<td>15</td>
<td>CENTR Data 6</td>
<td>16</td>
<td>GND</td>
</tr>
<tr>
<td>17</td>
<td>CENTR Data 7</td>
<td>18</td>
<td>GND</td>
</tr>
<tr>
<td>19</td>
<td>CENTR ACK</td>
<td>20</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>CENTR BSY</td>
<td>22</td>
<td>GND</td>
</tr>
<tr>
<td>23</td>
<td>CENTR PE</td>
<td>24</td>
<td>GND</td>
</tr>
<tr>
<td>25</td>
<td>CENTR SLCT</td>
<td>26</td>
<td>CENTR INIT</td>
</tr>
<tr>
<td>27</td>
<td>CENTR AF</td>
<td>28</td>
<td>CENTR ERR</td>
</tr>
<tr>
<td>29</td>
<td>N.C.</td>
<td>30</td>
<td>GND</td>
</tr>
<tr>
<td>31</td>
<td>GND</td>
<td>32</td>
<td>N.C.</td>
</tr>
<tr>
<td>33</td>
<td>N.C.</td>
<td>34</td>
<td>N.C.</td>
</tr>
<tr>
<td>35</td>
<td>N.C.</td>
<td>36</td>
<td>CENTR SLCT IN</td>
</tr>
<tr>
<td>37</td>
<td>N.C.</td>
<td>38</td>
<td>N.C.</td>
</tr>
<tr>
<td>39</td>
<td>N.C.</td>
<td>40</td>
<td>N.C.</td>
</tr>
</tbody>
</table>
### Table 20: IOBP-10 P5 Pinout (Serial)

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Signal</th>
<th>Pin No.</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>2</td>
<td>Keyboard In</td>
</tr>
<tr>
<td>3</td>
<td>Mouse In</td>
<td>4</td>
<td>Keyboard Out</td>
</tr>
<tr>
<td>5</td>
<td>TxD Port B</td>
<td>6</td>
<td>TxD Port A</td>
</tr>
<tr>
<td>7</td>
<td>RxD Port B</td>
<td>8</td>
<td>RxD Port A</td>
</tr>
<tr>
<td>9</td>
<td>RTS Port B</td>
<td>10</td>
<td>RTS Port A</td>
</tr>
<tr>
<td>11</td>
<td>CTS Port B</td>
<td>12</td>
<td>CTS Port A</td>
</tr>
<tr>
<td>13</td>
<td>GND</td>
<td>14</td>
<td>GND</td>
</tr>
</tbody>
</table>

### Table 21: IOBP-10 P6 Pinout (Ethernet)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>Collision+</td>
</tr>
<tr>
<td>3</td>
<td>Transmit Data+</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>Receive Data+</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>7</td>
<td>N.C.</td>
</tr>
<tr>
<td>8</td>
<td>N.C.</td>
</tr>
<tr>
<td>9</td>
<td>Collision-</td>
</tr>
<tr>
<td>10</td>
<td>Transmit Data-</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
</tr>
<tr>
<td>12</td>
<td>Receive Data-</td>
</tr>
<tr>
<td>13</td>
<td>+12VDC</td>
</tr>
<tr>
<td>14</td>
<td>GND</td>
</tr>
<tr>
<td>15</td>
<td>N.C.</td>
</tr>
</tbody>
</table>
2.8 How to Determine the Ethernet Address and Host ID

This information explains how an Ethernet number and a host ID number are determined on a CPU-5V board.

The 48-Bit (Six Byte) Ethernet Address

<table>
<thead>
<tr>
<th>Bytes</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

- The value of these three bytes always remains 008042 (HEX).
- Specific Machine (HEX)
- These two bytes contain the rightmost four digits of the board’s serial number in its hexadecimal representation.

The Ethernet Address

The Ethernet Address consists of a 48-bit (6-byte) number. The value of the most significant 24 bits is always 008042 (HEX). The value of the least significant 24 bits is calculated as follows.

- The bits 23 and 22 identify whether the product is designed by FORCE COMPUTERS GmbH or by FORCE COMPUTERS Inc. The value is 00 for GmbH products and the value is 01 for Inc. products.
- The bits 21 through 16 (least significant six bits of the third byte) identify a specific machine. The value of a CPU-5V board is 11 (decimal). Thus, the value of the most significant 32 bits (most significant four bytes) of a CPU-5V board from GmbH is 0080420B.
- The least significant 16 bits (least significant two bytes) contain the rightmost four digits of the board’s serial number in its hexadecimal representation.

Sample Serial Number: 2910 (decimal) = 0B5E (hexadecimal)

Sample Ethernet Address: 00 80 42 0B 0B 5E (hexadecimal)
The Host ID Number

The host ID is a 32-bit (4-byte) number and the eight most significant bits identify the architecture of the machine. The value representing the architecture of CPU-5V is 80 (HEX). The least significant 24 bits (least significant three bytes) contain the rightmost four digits of the board’s serial number in its hexadecimal representation and the value 8B7000 (HEX) must be added to the converted serial number.

The Host ID Number

<table>
<thead>
<tr>
<th>Bytes</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td>32</td>
<td>25</td>
<td>24</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

These 8 bits identify architecture type

The least significant 24 bits contain the rightmost four digits of the board’s serial number in its hexadecimal representation and the value 8B7000 must be added to the converted number.

Sample Serial Number: 2910 (decimal) --> 0B5E (hexadecimal)

\[
\begin{align*}
80 & \ 8B \ 7000 \\
+ & \ 0B5E \\
\hline
80 & \ 8B \ 7B5E \ (hexadecimal)
\end{align*}
\]

Sample Host ID: 80 8B 7B5E (hexadecimal)
SECTION 3 HARDWARE DESCRIPTION

3. Overview

Based on FORCE COMPUTERS FGA-5000 VMEbus to SBus interface gate array, the SPARC CPU-5V provides high speed VMEbus transfer capabilities for standard transfers and extended 64-bit MBLT transfers. In addition, the SPARC CPU-5V implements the capabilities of Sun Microsystems' SPARCstation 5 workstation on a single-slot VMEbus board.

The SPARC CPU-5V is powered by the microSPARC-II processor, which delivers a sustained processing performance of 76 SPECint92 and 65 SPECfp92. The complete suite of I/O functions includes fast SCSI-2, Ethernet, floppy disk, serial I/O, Centronics parallel I/O and keyboard/mouse ports making the SPARC CPU-5V the ideal solution for computing and VME transfer intensive embedded applications.

A complete 64-bit VMEbus interface and two standard SBus slots enable the expansion of I/O memory and processing performance with a broad range of off-the-shelf solutions. The software support for the SPARC CPU-5V ranges from Solaris, the most popular implementation of the UNIX operating system on a RISC architecture, to sophisticated real-time operating systems such as VxWorks.

The SPARC CPU-5V is a single board computer combining workstation performance and functionality with the ruggedness and expandability of an industry-standard single-slot 6U VMEbus board.

3.1 Block Diagram

A block diagram showing a functional overview of the CPU-5V is shown on the next page.
FIGURE 12. Block Diagram of the SPARC CPU-5V
3.2 The microSPARC-II Processor

The microSPARC-II CPU chip is at the core of the SPARC CPU-5V. This chip is realized in a 321-pin CPGA package. A Floating Point Unit, an Integer Unit, an MMU, an Instruction Cache, and a Data Cache are integrated in the microSPARC-II processor. Please see the microSPARC-II User’s Manual (STP1012PGA) for further information.

3.2.1 Features of the microSPARC-II Processor

• microSPARC-II chip running at 85 MHz and 110 MHz
• Integer Unit with 5-stage pipeline
• Floating Point Unit
• SPARC Reference Memory Management Unit
• A 16 Kbyte instruction cache and an 8 Kbyte data cache, directly mapped
• Memory interface which supports up to 256 Mbyte DRAM
• SBus controller supports up to five SBus slots plus one "master-only" slot
### 3.2.2 Address Mapping for microSPARC-II

The table below lists the physical addresses of the microSPARC-II processor.

**Table 22: Physical Memory Map of microSPARC-II**

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>SBus Slot #</th>
<th>Select #</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000 -&gt; 0FFFF FFFF</td>
<td>User Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000 0000 -&gt; 1FFFF FFFF</td>
<td>Control Space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000 0000 -&gt; 2FFFF FFFF</td>
<td>AFX Frame buffer</td>
<td>SBus Slot 0</td>
<td></td>
</tr>
<tr>
<td>3000 0000 -&gt; 3FFFF FFFF</td>
<td>SBus Slot 1</td>
<td></td>
<td>SBus Slave Select 0</td>
</tr>
<tr>
<td>4000 0000 -&gt; 4FFFF FFFF</td>
<td>SBus Slot 2</td>
<td></td>
<td>SBus Slave Select 1</td>
</tr>
<tr>
<td>5000 0000 -&gt; 5FFFF FFFF</td>
<td>SBus Slot 3</td>
<td></td>
<td>SBus Slave Select 2</td>
</tr>
<tr>
<td>6000 0000 -&gt; 6FFFF FFFF</td>
<td>SBus Slot 4</td>
<td></td>
<td>SBus Slave Select 3</td>
</tr>
<tr>
<td>7000 0000 -&gt; 7FFFF FFFF</td>
<td>SBus Slot 5</td>
<td></td>
<td>SBus Slave Select 4</td>
</tr>
</tbody>
</table>
3.3 The Shared Memory

The microSPARC-II chip interfaces directly to a 64-bit wide DRAM on one side and to the SBus on the other side. The shared DRAM is 64-bit wide with one parity bit for 32-bit data. The SPARC CPU-5V provides 16- or 64-Mbyte DRAM which is assembled on the board itself. There are 4-Mbit devices used to realize 16 Mbytes and there are 16-Mbit devices to realize 64 Mbytes.

The microSPARC-II chip supports up to eight memory banks (bank 0 to bank 7). Two of the eight memory banks are used on the SPARC CPU-5V base board (bank 0 and bank 1). The signals for the remaining memory banks are routed to the memory module connectors for module #1 and module #2.

Memory connector for memory module #1 supports banks 2, 3, 4 and 5. Memory connector for memory module #2 supports banks 4, 5, 6 and 7. Memory modules with up to 4 memory banks can be used. As shown in the table below, the memory bank structure is organized so that memory modules with a bank count from 1 to 4 (if available) can be used in any combination. Each module has up to 4 banks, only up to 8 banks in total are allowed. A memory module can contain bank A, or banks A and B, or banks A, B and C, or bank A, B, C and D.

Table 23: Bank Selection

<table>
<thead>
<tr>
<th>Bank Select from Processor</th>
<th>Base-board</th>
<th>Module on Connector #1</th>
<th>Module on Connector #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The shaded area above shows an example of how the banks are selected by the processor. In other words, the processor can select Bank B of the module on connector #1 by its own bank select 3. CAUTION: Do not connect more than one physical memory bank to one bank select from the processor. In other words, you can connect either bank C of the module on connector # 1, or bank D of module on connector # 2 to bank select 4, or you can connect either bank D of the module on connector # 1, or bank C of the module on connector # 2 to bank select 5.
The table below shows the base board memory capacity and the memory banks used by the microSPARC-II.

### Table 24: CPU-5V Memory Banks

<table>
<thead>
<tr>
<th>CPU-5V Memory Capacity</th>
<th>Memory Banks 0 and 1 are Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 Mbytes</td>
<td>X</td>
</tr>
<tr>
<td>64 Mbytes</td>
<td>X</td>
</tr>
</tbody>
</table>

### 3.4 Memory Module MEM-5

It is possible to upgrade your CPU-5V board with one or two memory modules (the remaining banks 2 to 7). The memory modules are available in different variants.

The MEM-5 provides 16- or 64-Mbyte DRAM. There are 4-Mbit devices used to realize 16 Mbytes and there are 16-Mbit devices to realize 64 Mbytes.

The table below shows the board memory capacity and the memory banks used on the microSPARC-II.

To understand the structure of the memory make sure you read "The Shared Memory" on page 55.

### Table 25: MEM-5 Memory Banks

<table>
<thead>
<tr>
<th>MEM-5 Memory Capacity</th>
<th>Memory Banks A and B are Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 Mbytes</td>
<td>X</td>
</tr>
<tr>
<td>64 Mbytes</td>
<td>X</td>
</tr>
</tbody>
</table>

Installing the memory modules is described in the document "How to Install MEM-5."
3.5 SBus Participants

There are two SBus slots located on the component side of the board. SBus Slot #2 is located at connector P3 and SBus Slot #3 is located at connector P4. A diagram of the board is located in the figure “Diagram of the CPU-5V (Top View)” on page 10.

The microSPARC-II chip supports up to 5 SBus slots plus an additional "master-only" slot. The SBus controller is inside the microSPARC-II chip.

The following table shows the microSPARC-II physical address map including all of its SBus slots and their functions on the SPARC CPU-5V.

3.5.1 Address Mapping for SBus Slots on the SPARC CPU-5V

Table 26: Physical Memory Map of SBus on SPARC CPU-5V

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>SBus Slot #</th>
<th>Select #</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000 0000 -&gt; 2FFFF FFFF</td>
<td>AFX Frame buffer</td>
<td>SBus Slot 0</td>
<td></td>
</tr>
<tr>
<td>3000 0000 -&gt; 3FFFF FFFF</td>
<td>VMEbus Interface</td>
<td>SBus Slot 1</td>
<td>SBus Slave Select 0</td>
</tr>
<tr>
<td>4000 0000 -&gt; 4FFFF FFFF</td>
<td>SBus Module P3</td>
<td>SBus Slot 2</td>
<td>SBus Slave Select 1</td>
</tr>
<tr>
<td>5000 0000 -&gt; 5FFFF FFFF</td>
<td>VMEbus Interface</td>
<td>SBus Slot 3</td>
<td>SBus Slave Select 2</td>
</tr>
<tr>
<td>6000 0000 -&gt; 6FFFF FFFF</td>
<td>VMEbus Interface</td>
<td>SBus Slot 4</td>
<td>SBus Slave Select 3</td>
</tr>
<tr>
<td>7000 0000 -&gt; 7FFFF FFFF</td>
<td>NCR89C105 (SLAVIO) chip</td>
<td>SBus Slot 5</td>
<td>SBus Slave Select 4</td>
</tr>
<tr>
<td>7800 0000 -&gt; 7DFFFF FFFF</td>
<td>NCR89C100 (MACIO) chip</td>
<td>SBus Slot 5</td>
<td>SBus Slave Select 4</td>
</tr>
<tr>
<td>7E00 0000 -&gt; 7FFFF FFFF</td>
<td>VMEbus Interface</td>
<td>SBus Slot 5</td>
<td>SBus Slave Select 4</td>
</tr>
</tbody>
</table>

If there are no SBus modules installed, SBus Slot 1, 2, 3 and 4, together with SBus Slot 5 address range: 7E00 0000-7FFFF FFFF, are available for the VMEbus Interface.
3.6 NCR89C100 (MACIO)

The NCR89C100 is located on SBus Slave Select 4 at physical address $7800 0000. This chip drives the SCSI, Ethernet and Centronics parallel port.

The NCR89C100 SBus master integrates high performance I/O macrocells and logic including an Ethernet controller core, a fast 53C9X SCSI core, a high-speed parallel port, a DMA2 controller and an SBus interface.

The Ethernet core is compatible with the industry standard 7990 Ethernet controller. The SCSI core is a superset of the industry standard NCR53C90A which has been modified to support fast SCSI. The uni/bi-directional parallel port is Centronics compliant and can operate in either programmed I/O or DMA mode.

The DMA2 block comprises the logic used to interface each of these functions to the SBus. It provides buffering for each of the functions. Buffering takes the form of a 64-byte data cache and 16-bit wide buffer for the Ethernet channel, and a 64-byte FIFO for both the SCSI channel and the parallel port. The DMA2 incorporates as improved cache and FIFO draining algorithm which allows better SBus utilization than previous DMA implementations.
3.6.1 Features of the NCR89C100 on the SPARC CPU-5V

- Fast 8-bit SCSI
  - Supports fast SCSI mode
  - Backward compatible to 53C90A
- 7990-compatible Ethernet
- Parallel Port
  - I/O or DMA programmable modes
  - Centronics compatibility
- LS64854-compatible DMA2 Controller
- Glueless SBus Interface clocked with 21.25 MHz @ 85 MHz processor frequency
- Glueless SBus Interface clocked with 22.00 MHz @ 110 MHz processor frequency
- Concurrently supports:
  - 10 MB/sec SCSI transfers
  - 3.4 MB/sec Parallel port transfers
  - 1.25 MB/sec Ethernet transfers
- 64-byte FIFO for SCSI and Parallel Port data
- Supports SBus burst modes
  - 4-word, 8-word and “no/burst”
- Packaged in 160-pin PQFP

For further information about the NCR89100, please see NCR SBus I/O Chipset Data Manual.
3.6.2 SCSI

The SCSI interface provides a standard interface to a wide variety of mass storage devices, such as hard disks, tapes and CD-ROMs. The SCSI transfers up to 10 Mbytes per second.

The SPARC CPU-5V board's SCSI is realized via the NCR89C100. The NCR89C100 has on-chip 48 mA drivers and therefore provides direct drive of single-ended SCSI bus. The SCSI core is a superset of the industry standard NCR53C90A which has been modified to support fast SCSI.

The SCSI interface is single ended and supports "TERMPWR". The NCR89C100 DMA2 core is able to transfer the data to and from the shared main memory.

All signals of the SCSI interface are routed to a standard connector on the front panel and to the VME P2 connector. This 2nd connection is compatible to the CPU-2CE, CPU-3CE and the CPU-5CE. Please see the "VME P2 Connector Pinout" on page 41 where the SCSI signals on the VME P2 connector are shown.

3.6.3 SCSI Termination

When only one of the SCSI connectors is used, the other one is an end point. In that case, the SCSI bus must be terminated near the unused connector. This is supported on the CPU-5V board through one termination at the front panel and one termination at VME P2.

The front panel termination of the SCSI can be configured via the on-board switch SW6-1. If SW6-1 is ON, the front panel termination is disabled. If SW6-1 is OFF, the termination is automatic. Automatic means that when a SCSI cable is plugged into the front panel connector, the termination is automatically disabled. When there is no SCSI cable plugged into the front panel, the termination is automatically enabled.

The VME P2 termination of the SCSI interface can be enabled or disabled via the switch SW6-2. If SW6-2 is OFF, the termination is enabled. If SW6-2 is ON, the termination is disabled.

Please see "Diagram of the CPU-5V (Top View)" on page 10 for the location of the switches on the board.

CAUTION

When installing the SPARC CPU-5V in a MICROFORCE chassis, please first disable the SCSI termination by switching SW6-1 and SW6-2 to ON.
3.6.4 Ethernet

The NCR89C100 DMA controller enables the Ethernet interface to transfer data to and from the shared main memory. The Ethernet core is register level compatible with the AMD Am7990, Revision F, standard Ethernet controller, which is capable of transferring Ethernet data up to 10 Mbit/sec.

An 8-pin configuration switch matrix selects whether the Ethernet interface is available via the front panel or the VME P2 connector. By default, the Ethernet Interface is available through the front panel connector with the I/O bridge plugged into connectors B12 and B13. It is possible to have the Ethernet interface accessible on the VME P2 connector by changing the default configuration. In order that the Ethernet interface is accessible from the VMEbus P2 connector, the I/O bridge array must be plugged into connectors B11 and B12.

Please see the figure “Ethernet Interface via Front Panel” on page 19 for information about changing the Ethernet configuration.

CAUTION

When the Ethernet is configured via P2, do not connect the Ethernet at the front panel.

3.6.5 Parallel Port

The parallel port is centronics compliant and provides uni/bi-directional communication. It operates in either programmed I/O or DMA mode.

The default configuration enables the floppy interface via the VME P2 connector, with the configuration switch matrix plugged into B1 and B2. This means, of course, that by default the parallel port interface is not available via the VMEbus P2 connector. It is the floppy disk interface which is available on the VMEbus P2 connector by default.

In order to configure the parallel port to be accessible from the VMEbus P2 connector, the switch matrix must be plugged into connectors B2 and B3. Please see the figure “Floppy Interface Via VME P2 Connector” on page 18 for information about changing the configuration.
3.7 NCR89C105 (SLAVIO)

The NCR89C105 SBus slave integrates most of the 8-bit system I/O functions including two dual channel 8530-compatible serial controllers, a high speed 8277AA-1-compatible floppy disk controller, counter/timers, interrupt controllers, and system reset logic. It also provides an SBus interface for several other byte-wide peripherals through an external expansion bus.

The primary serial controller is 8530-compatible and can be used as two general purpose serial ports.

The second serial controller is subset of the 8530 standard and is dedicated for the keyboard/mouse connection.

The 8277AA-1-compatible floppy disk controller supports up to 1 Mbit/sec data transfer rate.

To reduce part count and system cost, a glueless interface to the SBus is provided. The slave I/O also includes an 8-bit expansion bus with control to support RTC/NVRAM, EPROM and generic 8-bit devices externally.

3.7.1 Features of the NCR89C105 on the SPARC CPU-5V

- Dual-channel serial ports (8530-compatible)
- Keyboard/mouse port
- 82077AA-1 floppy disk controller
- 8-bit expansion bus for Flash Memory/TOD/NVRAM
- Glueless SBus interface clocked with 21.25 MHz @ 85 MHz and 22.0 MHz @ 110 MHz processor frequency
- Interrupt controller
- System reset control
- Programmable 22-bit counters & timers
- Auxiliary I/O registers
- Packaged in 160-pin PQFP

For further information about the NCR89100, please refer to the NCR SBus I/O Chipset Data Manual.
3.7.2 Address Map of Local I/O Devices on SPARC CPU-5V

The following table lists the physical addresses for all local I/O devices and the accesses permitted ((B)yte, (H)alf Word and (W)ord).

Table 27: NCR89C105 Chip Address Map

<table>
<thead>
<tr>
<th>Physical Address</th>
<th>Device</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>7000 0000 -&gt;</td>
<td>Boot Flash Memory and User Flash Memory</td>
<td>B,H,W</td>
</tr>
<tr>
<td>70FF FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7100 0000 -&gt;</td>
<td>Keyboard, Mouse, and Serial Ports</td>
<td>B</td>
</tr>
<tr>
<td>711F FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7100 0000</td>
<td>Mouse Control Port</td>
<td></td>
</tr>
<tr>
<td>7100 0002</td>
<td>Mouse Data Port</td>
<td></td>
</tr>
<tr>
<td>7100 0004</td>
<td>Keyboard Control Port</td>
<td></td>
</tr>
<tr>
<td>7100 0006</td>
<td>Keyboard Data Port</td>
<td></td>
</tr>
<tr>
<td>7110 0000</td>
<td>TTYB Control Port</td>
<td></td>
</tr>
<tr>
<td>7110 0002</td>
<td>TTYB Data Port</td>
<td></td>
</tr>
<tr>
<td>7110 0004</td>
<td>TTYA Control Port</td>
<td></td>
</tr>
<tr>
<td>7110 0006</td>
<td>TTYA DATA Port</td>
<td></td>
</tr>
<tr>
<td>7120 0000 -&gt;</td>
<td>RTCNVRAM</td>
<td>B,H,W</td>
</tr>
<tr>
<td>712F FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7130 0000 -&gt;</td>
<td>Boot Flash Memory and User Flash Memory Programming</td>
<td>B</td>
</tr>
<tr>
<td>7137 FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7138 0000 -&gt;</td>
<td>Additional Registers</td>
<td>B</td>
</tr>
<tr>
<td>713F FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7140 0000 -&gt;</td>
<td>Floppy Controller</td>
<td>B</td>
</tr>
<tr>
<td>714F FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7140 0002</td>
<td>Digital Output Register (DOR)</td>
<td></td>
</tr>
<tr>
<td>7140 0004</td>
<td>Main Status Register (MSR, Read Only)</td>
<td></td>
</tr>
<tr>
<td>7140 0004</td>
<td>DateRate Select Register (DSR, Write Only)</td>
<td></td>
</tr>
<tr>
<td>7140 0005</td>
<td>FIFO</td>
<td></td>
</tr>
<tr>
<td>7140 0006</td>
<td>Reserved (Test mode select)</td>
<td></td>
</tr>
<tr>
<td>7140 0007</td>
<td>Digital Input Register (DIR, Read Only)</td>
<td></td>
</tr>
<tr>
<td>7140 0007</td>
<td>Configuration Control Register (CCR, Write Only)</td>
<td></td>
</tr>
<tr>
<td>7150 0000 -&gt;</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7170 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7180 0000</td>
<td>89C105 Configuration Register</td>
<td>B</td>
</tr>
<tr>
<td>7190 0000 -&gt;</td>
<td>Auxiliary I/O Registers</td>
<td>B</td>
</tr>
<tr>
<td>719F FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7190 0000</td>
<td>Aux 1 Register (Miscellaneous System Functions)</td>
<td></td>
</tr>
<tr>
<td>7191 0000</td>
<td>Aux 2 Register (Software Powerdown Control)</td>
<td></td>
</tr>
</tbody>
</table>
3.7.3 Serial I/O Ports

The two serial I/O ports are available on the front panel via one 26-pin shielded connector.

Both of the two ports are available via the VMEbus P2 connector, each with four signals (RXD, TXD, RTS, CTS). Each of the two serial I/O ports are independent full-duplex ports.

The 8530 SCC block is functionally compatible with the standard NMOS 8530 and therefore provides two fully independent full-duplex ports.

The physical address map for the serial ports is shown in “NCR89C105 Chip Address Map” on page 63.

3.7.4 RS-232, RS-422 or RS-485 Configuration

Both serial ports can be configured as RS-232, RS-422 or RS-485. By default, the FH-002 hybrid module is installed for RS-232 operation.

In order to simplify changing the serial interfaces, FORCE COMPUTERS has developed RS-232, RS-422 and RS-485 hybrid modules: the FH-002, FH-003 and FH-005. These 21-pin SIL modules are installed in sockets so that they may be easily changed to meet specific application needs.

To change the configuration of serial port A, insert the respective hybrid in socket J59. To change the configuration of serial port B, insert the respective hybrid in socket J60. For the position of the sockets on the board, please see “Diagram of the CPU-5V (Top View)” on page 10.

Table 28: RS-232, RS-422 or RS-485 Configuration

<table>
<thead>
<tr>
<th>Hybrid</th>
<th>Configuration</th>
<th>Socket for Serial Port A</th>
<th>Socket for Serial Port B</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>FH-002</td>
<td>RS-232</td>
<td>J59</td>
<td>J60</td>
<td>*</td>
</tr>
<tr>
<td>FH-003</td>
<td>RS-422</td>
<td>J59</td>
<td>J60</td>
<td></td>
</tr>
<tr>
<td>FH-005</td>
<td>RS-485</td>
<td>J59</td>
<td>J60</td>
<td></td>
</tr>
</tbody>
</table>
3.7.5 RS-232 Hardware Configuration

The serial ports A and B are configured by default for RS-232 operation. The following individual I/O signals are available for serial ports A and B on the front panel connector.

Table 29: Serial Ports A and B Pinout List (RS-232)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Transmitted Signals</th>
<th>Pin</th>
<th>Received Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>TXD-Transmit Data</td>
<td>3</td>
<td>RXD-Receive Data</td>
</tr>
<tr>
<td>4</td>
<td>RTS-Request to Send</td>
<td>5</td>
<td>CTS-Clear to Send</td>
</tr>
<tr>
<td>7</td>
<td>Ground</td>
<td>6</td>
<td>SYNC</td>
</tr>
<tr>
<td>20</td>
<td>DTR-Data Terminal Ready</td>
<td>8</td>
<td>DCD-Data Carrier Detect</td>
</tr>
<tr>
<td>24</td>
<td>TRXC-DTE Transmit Clock</td>
<td>15</td>
<td>TRXD-DCE Transmit Clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>RTXC-DCE Receive Clock</td>
</tr>
</tbody>
</table>

The pinout for serial port A is shown in the white area and the pinout for serial port B is shown in the grey area.

The table below shows the switch settings for each port.

Table 30: Switch Settings for Ports A and B (RS-232)

<table>
<thead>
<tr>
<th>Port A</th>
<th>Port B</th>
<th>Default</th>
<th>Function for RS-232</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW4-1</td>
<td>SW5-1</td>
<td>ON</td>
<td>TRXC is available on front panel connectors, pin 24</td>
</tr>
<tr>
<td>SW4-3</td>
<td>SW5-3</td>
<td>OFF</td>
<td>RTS is available on front panel connectors, pin 4</td>
</tr>
<tr>
<td>SW4-2</td>
<td>SW5-2</td>
<td>OFF</td>
<td>CTS is available on front panel connectors, pin 5</td>
</tr>
</tbody>
</table>

Please see the “Diagram of the CPU-5V (Bottom View)” on page 11 for the location of the switches on the board.
3.7.6 RS-422 Hardware Configuration

It is possible to reconfigure serial ports A and B for RS-422 operation. In order to configure the serial ports to RS-422, the hybrid module FH-003 must be used. Termination resistors can be installed to adapt various cable lengths and reduce reflections.

Table 31: Serial Ports A and B Pinout List (RS-422)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Transmitted Signals</th>
<th>Pin</th>
<th>Received Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>TXD+ Transmit Data</td>
<td>20</td>
<td>RXD+ Receive Data</td>
</tr>
<tr>
<td>8</td>
<td>TXD- Transmit Data</td>
<td>7</td>
<td>RXD- Receive Data</td>
</tr>
<tr>
<td>4*</td>
<td>RTS+ Request to Send</td>
<td>2*</td>
<td>CTS+ Clear to Send</td>
</tr>
<tr>
<td>3*</td>
<td>RTS- Request to Send</td>
<td>5*</td>
<td>CTS- Clear to Send</td>
</tr>
<tr>
<td>4*</td>
<td>TRXC+ Transmit Clock</td>
<td>2*</td>
<td>RTXC+ Receive Clock</td>
</tr>
<tr>
<td>3*</td>
<td>TRXC- Transmit Clock</td>
<td>5*</td>
<td>RTXC- Receive Clock</td>
</tr>
</tbody>
</table>

The pinout for serial port A is shown in the white area and the pinout for serial port B is shown in the grey area.

* Signals RTS and TRXC can be switched so that they are available on connector pins 3 and 4 (16, 19). Signals CTS and RTXC can also be switched so that they are available on connector pins 2 and 5 (14, 93). This is done by switch SW4 for port A and by switch SW5 for port B.

The table on the next page shows the corresponding switch settings.
Table 32: Switch Settings for Ports A and B (RS-422)

<table>
<thead>
<tr>
<th>Port A</th>
<th>Port B</th>
<th>Configuration</th>
<th>Function for RS-422</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW4-1</td>
<td>SW5-1</td>
<td>ON</td>
<td>ON for RS-422</td>
</tr>
<tr>
<td>SW4-3</td>
<td>SW5-3</td>
<td>ON</td>
<td>TRXC +/- on front panel connectors, pins 3/16 and 4/19 available</td>
</tr>
<tr>
<td>SW4-3</td>
<td>SW5-3</td>
<td>OFF</td>
<td>RTS +/- on front panel connectors, pins 3/16 and 4/19 available</td>
</tr>
<tr>
<td>SW4-2</td>
<td>SW5-2</td>
<td>OFF</td>
<td>CTS +/- on front panel connectors, pins 2/14 and 5/13 available</td>
</tr>
<tr>
<td>SW4-2</td>
<td>SW5-2</td>
<td>ON</td>
<td>RTXC +/- on front panel connectors, pins 2/14 and 5/13 available</td>
</tr>
</tbody>
</table>

Please see the “Diagram of the CPU-5V (Bottom View)” on page 11 for the location of the switches on the board.
3.7.7 RS-485 Hardware Configuration

It is possible to reconfigure serial ports A and B to be RS-485 compatible by using the hybrid module FH-OOS.

The following I/O signals are available on the front panel connectors of both serial ports.

Table 33: Serial Ports A and B Pinout List (RS-485)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>23 RXTX+ Receive/Transmit Data</td>
</tr>
<tr>
<td>20</td>
<td>11 RXTX- Receive/Transmit Data</td>
</tr>
</tbody>
</table>

The pinout for serial port A is shown in the white area and the pinout for serial port B is shown in the grey area.

The Receive-Enable (REN) and Transmit-Enable (TEN) of the hybrid module FH-OOS are controlled via the NCR89C105 serial I/O signals DTR (REN) and RTS(TEN).

The following table shows the corresponding switch settings

Table 34: Switch Settings for Ports A and B (RS-485)

<table>
<thead>
<tr>
<th>Port A</th>
<th>Port B</th>
<th>Configuration</th>
<th>Function for RS-485</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW4-1</td>
<td>SW5-1</td>
<td>OFF</td>
<td>RTS functions as TEN</td>
</tr>
<tr>
<td>SW4-3</td>
<td>SW5-3</td>
<td>OFF</td>
<td>No function for RS-485</td>
</tr>
<tr>
<td>SW4-2</td>
<td>SW5-2</td>
<td>OFF</td>
<td>No function for RS-485</td>
</tr>
</tbody>
</table>

Please see the “Diagram of the CPU-5V (Bottom View)” on page 11 for the location of the switches on the board.
3.7.8 Keyboard and Mouse Port

The keyboard and mouse port is available on the front panel via an 8-pin mini DIN connector and on VME P2.

The serial port controller used for the keyboard and mouse port is compatible with the NMOS 8530 controller.

The pinout of the keyboard and mouse port is described in Section 2, Installation.

The physical address for the keyboard and mouse port is shown in “NCR89C105 Chip Address Map” on page 63.

3.7.9 Floppy Disk Interface

The floppy disk interface is available on the P2 connector by default, with the configuration switch matrix plugged into B1 and B2. The floppy disk configuration is described in the chapter “Parallel Port or Floppy Interface via VME P2 Connector” on page 18.

The floppy disk interface is 82077AA-1 compatible. It is able to transfer data rates of 250, 300, 500 Kbytes/sec, and 1 Mbyte/sec.

The floppy disk controller block is functionally compatible with the Intel 82077AA-1. It integrates drivers, receivers, data separator, and a 16-byte bidirectional FIFO. The floppy disk controller supports all standard disk formats (typically 720 K and 1.44 M floppies). It is also compatible with the 2.88 MB floppy format.
3.7.10 8-Bit Local I/O Devices

The following local I/O devices are interfaced via the NCR89C105

Table 35: 8-Bit Local I/O Devices

<table>
<thead>
<tr>
<th>Function</th>
<th>IRQ</th>
<th>Physical Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot Flash Memory Device # 1</td>
<td>No</td>
<td>$7000 0000 -&gt; $7003 FFFF</td>
</tr>
<tr>
<td>256 Kbyte (default)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boot Flash Memory Device # 2</td>
<td>No</td>
<td>$7004 0000 -&gt; $7007 FFFF</td>
</tr>
<tr>
<td>256 Kbyte (default)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Flash Memory 1 Mbyte Device # 1</td>
<td>No</td>
<td>$7010 0000 -&gt; $701F FFFF</td>
</tr>
<tr>
<td>User Flash Memory 1 Mbyte Device # 2</td>
<td>No</td>
<td>$7020 0000 -&gt; $702F FFFF</td>
</tr>
<tr>
<td>RTC/NVRAM</td>
<td>No</td>
<td>$7120 0000 -&gt; $712F FFFF</td>
</tr>
<tr>
<td>Flash Memory Programming Area</td>
<td>No</td>
<td>$7130 0000 -&gt; $7137 FFFF</td>
</tr>
<tr>
<td>Additional Registers</td>
<td>No</td>
<td>$7138 0000 -&gt; $713F FFFF</td>
</tr>
</tbody>
</table>
3.7.11 Boot Flash Memory

The boot flash memory consists of two 2-Mbit or 4-Mbit flash memory devices. In the default configuration, there are two 2-Mbit devices installed. The 4-Mbit devices are an additional assembly option.

The boot flash memory devices can be reprogrammed on-board and can also be write protected via hardware switch SW6-3.

When SW6-3 is ON, write accesses are possible. The devices are write protected when SW6-3 is OFF.

The boot flash memory devices are installed in sockets at location J26 (device #1) and J22 (device #2). This permits programming them in a standard programmer. This may be necessary if the power fails during reprogramming. In this case, the contents of the Boot Flash Memory would be lost and the board would not be able to boot.

<table>
<thead>
<tr>
<th>Devices</th>
<th>Count</th>
<th>Capacity</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 K * 8</td>
<td>2</td>
<td>512 Kbyte</td>
<td>X</td>
</tr>
<tr>
<td>512 K* 8</td>
<td>2</td>
<td>1 Mbyte</td>
<td></td>
</tr>
</tbody>
</table>

The on-board programming of the boot flash memory devices requires setting some bits in the Flash Memory Programming Voltage Control Register and Flash Memory Programming Control Register 1 and 2. These registers are shown on the following pages.
3.7.12 User Flash Memory

The user flash memory area consists of a maximum of two 8-Mbit flash memory devices, providing a capacity of 2 Mbytes. The capacity of user flash memory is outlined in the product nomenclature, which can be seen in the table “Ordering Information” on page 6.

This area can be used to store ROMable operating systems as well as application specific code.

<table>
<thead>
<tr>
<th>Devices</th>
<th>Count</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M*8</td>
<td>2</td>
<td>2 Mbyte</td>
</tr>
</tbody>
</table>

The user flash memory devices can be reprogrammed on-board and can also be write protected via hardware switch SW6-4. When SW6-4 is ON, write accesses are possible. When SW6-4 is OFF, the devices are write protected.

The on-board programming of the user flash memory devices requires setting some bits in the Flash Memory Programming Voltage Control Register and Flash Memory Programming Control Register 1 and 2. These registers are shown on the following pages.
3.7.13 Programming the On-board Flash Memories

Both areas of flash memories, the Boot area and the User area, can be reprogrammed on-board. Please see “Flash Memory Support” on page 162 for details about programming the on-board memories.

The address range in which the flash memory devices can be programmed is located in a 512 Kbyte page (programming window) of the Generic Port area of the NCR89C105 (SLAVIO). The physical address range is $7130 0000 .. $7137 FFFF.

Please note the following steps for programming the on-board flash memory devices.

- Disable hardware write protection in order to program the flash memory devices. The switch SW6-3 must be ON in order to program the boot flash memory and the switch SW6-4 must be ON in order to program the user flash memory. For the location of the switches on the board please see “Diagram of the CPU-5V (Bottom View)” on page 11.
- Switch the programming voltage ON by setting the appropriate bit.
- Set address lines A[21:19] to the requested address range. Set the device number of the device to be selected.
- Select either the user flash memory or the boot flash memory for programming.
- After the flash memory devices have been programmed, we recommend that you return to the default settings of SW6-3 and SW6-4. This protects the flash memory devices from being programmed by accident.

In order to enable programming and to decide which area is to be mapped to the programming window, the following six bits VPP_ON, A[21:19], SEL_ROM and SEL_BOOT are used to control this.
3.7.13.1 Flash Memory Programming Voltage Control Register

To enable the programming of the flash memory devices, the +12V programming voltage must be switched ON. This is done by setting bit VPP ON in the Flash Memory Programming Voltage Control Register.

**Initialization**: VPP ON is cleared on reset. This inhibits the programming of the flash memory devices.

**Physical Address**: 7138 000A<sub>16</sub>

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>VPP ON</td>
</tr>
</tbody>
</table>

VPP ON (RW) This bit is used to turn the +12V programming voltage for all flash memories on or off. When the bit is set (1) then the programming voltage is turned on; and the programming voltage is turned off by clearing (0) this bit.
3.7.13.2  Flash Memory Programming Control Register 1

Physical Address: 7138 000216

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0002</td>
<td>0</td>
<td>00.0000_16 ... 07.FFFF_16</td>
</tr>
<tr>
<td>0012</td>
<td>1</td>
<td>08.0000_16 ... 0F.FFFF_16</td>
</tr>
<tr>
<td>0102</td>
<td>2</td>
<td>10.0000_16 ... 17.FFFF_16</td>
</tr>
<tr>
<td>0112</td>
<td>3</td>
<td>18.0000_16 ... 1F.FFFF_16</td>
</tr>
<tr>
<td>1002</td>
<td>4</td>
<td>20.0000_16 ... 27.FFFF_16</td>
</tr>
<tr>
<td>1012</td>
<td>5</td>
<td>28.0000_16 ... 2F.FFFF_16</td>
</tr>
<tr>
<td>1102</td>
<td>6</td>
<td>30.0000_16 ... 37.FFFF_16</td>
</tr>
<tr>
<td>1112</td>
<td>7</td>
<td>38.0000_16 ... 3F.FFFF_16</td>
</tr>
</tbody>
</table>

SEL_ROM (RW) This bit and the SEL_BOOT bit in the Flash Memory Programming Control Register 2 are used to select one of four flash memory devices to be accessible in the physical address range 7130.0000_16 to 7137.FFFF_16.
3.7.13.3 Flash Memory Programming Control Register 2

Physical Address: 7138 0009_{16}

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>SEL_BOOT</td>
</tr>
</tbody>
</table>

SEL_BOOT (RW) This bit and the SEL_ROM bit in the Flash Memory Programming Control Register 1 are used to select one of four flash memories to be accessible in the physical address range 7130.0000_{16} to 7137.FFFF_{16}. The following table shows all possible configurations:

<table>
<thead>
<tr>
<th>SEL_BOOT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SEL_ROM</td>
<td>first USER flash memory device accessible</td>
<td>first BOOT flash memory device accessible</td>
</tr>
<tr>
<td>1</td>
<td>second USER flash memory device accessible</td>
<td>second BOOT flash memory device accessible</td>
</tr>
</tbody>
</table>
3.7.14 RTC/NVRAM

The MK48T08 combines an 8 K x 8 full CMOS SRAM, a bytewide accessible Real Time Clock, a crystal, and a long-life lithium carbon monofluoride battery, all in a single plastic DIP package. The MK48T08 is a nonvolatile pin and functionally equivalent to any Jede standard 8 K x 8 SRAM.

For a detailed description of the RTC/NVRAM, please see the respective Data Sheet.
3.8 VMEbus Interface

The CPU-5V utilizes the FGA-5000 chip to provide fully SBus and VMEbus compliant interfaces. Supported functions include master and slave data transfer capabilities, VMEbus interrupt handling and arbitration functions. Additional VMEbus utility functions and a special loop-back cycle for stand-alone testing of the interface are provided.

Features of the FGA-5000

- VMEbus Master Interface
- VMEbus Slave Interface
- DMA Controller
- Interrupts
- VMEbus Arbiter
- FORCE Message Broadcast
- Mailboxes and Semaphores
- Reset Functions
- System Controller Functions
- Timers

A complete description of the FGA-5000 chip is found in the FGA-5000 Technical Reference Manual, available from FORCE COMPUTERS.
3.8.1 Address Mapping for the VMEbus Interface FGA-5000

The table below lists the physical addresses of the VMEbus interface FGA-5000.

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>SBus Slot #</th>
<th>Select #</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000 0000 -&gt;</td>
<td>VMEbus Interface</td>
<td>SBus Slot 1</td>
<td>SBus Slave Select 0</td>
</tr>
<tr>
<td>3FFFFFFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000 0000 -&gt;</td>
<td>VMEbus Interface (SBus Module)</td>
<td>SBus Slot 2</td>
<td>SBus Slave Select 1</td>
</tr>
<tr>
<td>4FFFFFFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000 0000 -&gt;</td>
<td>VMEbus Interface (SBus Module)</td>
<td>SBus Slot 3</td>
<td>SBus Slave Select 2</td>
</tr>
<tr>
<td>5FFFFFFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6000 0000 -&gt;</td>
<td>VMEbus Interface</td>
<td>SBus Slot 4</td>
<td>SBus Slave Select 3</td>
</tr>
<tr>
<td>6FFFFFFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7000 0000 -&gt;</td>
<td>VMEbus Interface</td>
<td>SBus Slot 5</td>
<td>SBus Slave Select 4</td>
</tr>
<tr>
<td>7FFFFFFF</td>
<td>(SBus Slave Select 4)</td>
<td></td>
<td>(SB_SEL&lt;5&gt;)</td>
</tr>
</tbody>
</table>

The FGA-5000 can be selected with up to six SBus select input signals SSEL<5..0>. The microSPARC-II CPU chip supports only 5 select signals SLVSEL<4..0>. The remaining select signal of the FGA-5000 can be used to expand the VMEbus address area.

The I/O chips NCR89C100 (MACIO) and NCR89C105 (SLAVIO) are selected in SBus Slot 5 by SBus Slave Select 4. The upper part in this range is not used by these chips. So we decided to split the SBus Slave Select 4 into two signals: SB_SEL<4> and SB_SEL<5>. Now the I/O chips are selected by SB_SEL<4> and the VMEbus interface FGA-5000 is selected by SB_SEL<5>. With this expansion, the VMEbus interface FGA-5000 shares SBus Slot 5 with the I/O chips and can use an additional address range of up to 32 Mbyte in this SBus Slot.

On the base board, SBus Slot 2 (SBus Slave Select 1) and SBus Slot 3 (SBus Slave Select 2) are provided for SBus modules. When no SBus modules are installed, you can use these SBus slots to expand the VMEbus address range again. In this case, you gain 256 Mbyte with every additional SBus Slot.

When using all address range resources for the VMEbus interface the microSPARC-II CPU can access the VMEbus interface FGA-5000, internal registers and VMEbus slaves, in an address area of 1056 Mbyte (1GByte + 32 Mbyte).
3.8.2 Adaptation of the FGA-5000

Some aspects of the VMEbus interface chip FGA-5000 require a small amount of glue logic be built around this chip in order to use the chip on this base board.

In the case where the FGA-5000 is not VMEbus master, the VMEbus input signal BERR has not been directly routed to the FGA-5000. This masking is required for normal VMEbus transfers. However, during FMB transfers the VMEbus slave should see the input signal BERR.

When the FGA-5000 is one of several selected slaves during an FMB cycle, and one or more of the other slaves acknowledges the transfer with Bus Error, then the FGA-5000 doesn’t recognize that the message is invalid. In order to enable the software to discard this invalid message, additional registers have been implemented in a separate programmable device on the base board.

How to access and interpret the contents of the added registers can be found in the chapter “Additional Registers” on page 90.

For information about the FMB implementation in the FGA-5000 chip, please refer to the FGA-5000 Technical Reference Manual.
3.8.3 VMEbus SYSRESET Enable/Disable

3.8.3.1 SYSRESET Input

An external SYSRESET generates an on-board RESET in the default switch setting, i.e., SW7-3 is ON. When SW7-3 is OFF, the external SYSRESET does not generate an on-board RESET.

3.8.3.2 SYSRESET Output

An on-board RESET drives the SYSRESET signal to the VMEbus to low in the default switch setting, i.e., SW7-4 is ON. When SW7-4 is OFF, an on-board RESET doesn’t drive the SYSRESET signal to the VMEbus to low.

CAUTION

Do not switch SW7-4 (SYSRESET output) to ON and SW8-2 (VMEbus Slot-1 device) to OFF at the same time.

The VMEbus Specification requires that if SYSRESET is driven, the SYSRESET signal shall be driven low for at least 200 ms. However, when the CPU-5V is not a VMEbus Slot-1 device and the SYSRESET output signal is enabled, then the CPU-5V no longer conforms with this rule.

By default, the SYSRESET output is enabled. In this case it generates the SYSRESET signal to the VMEbus.
3.9 On-board Control Registers (System Configuration)

The following table shows the physical address of the registers used for system configuration. The registers are described in their respective functional chapters, for example, the USER LEDs are described in the chapter “Front Panel Status LEDs” on page 85.

<table>
<thead>
<tr>
<th>Address</th>
<th>Reset Value</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7138.0000_{16}</td>
<td>F0_{16}</td>
<td>8 bit</td>
<td>USER LED 1 Control Register</td>
</tr>
<tr>
<td>7138.0001_{16}</td>
<td>F0_{16}</td>
<td>8 bit</td>
<td>USER LED 2 Control Register</td>
</tr>
<tr>
<td>7138.0002_{16}</td>
<td>F0_{16}</td>
<td>8 bit</td>
<td>Flash Memory Programming Control Register 1</td>
</tr>
<tr>
<td>7138.0003_{16}</td>
<td>FX_{16}</td>
<td>8 bit</td>
<td>Rotary Switch Status Register</td>
</tr>
<tr>
<td>7138.0004_{16}</td>
<td>XX_{16}</td>
<td>8 bit</td>
<td>Reserved</td>
</tr>
<tr>
<td>7138.0005_{16}</td>
<td>XX_{16}</td>
<td>8 bit</td>
<td>Reserved</td>
</tr>
<tr>
<td>7138.0006_{16}</td>
<td>XX_{16}</td>
<td>8 bit</td>
<td>Reserved</td>
</tr>
<tr>
<td>7138.0007_{16}</td>
<td>XX_{16}</td>
<td>8 bit</td>
<td>Reserved</td>
</tr>
<tr>
<td>7138.0008_{16}</td>
<td>FE_{16}</td>
<td>8 bit</td>
<td>Boot ROM Size Control Register</td>
</tr>
<tr>
<td>7138.0009_{16}</td>
<td>FE_{16}</td>
<td>8 bit</td>
<td>Flash Memory Programming Control Register 2</td>
</tr>
<tr>
<td>7138.000A_{16}</td>
<td>FE_{16}</td>
<td>8 bit</td>
<td>Flash Memory Programming Voltage Control Register</td>
</tr>
<tr>
<td>7138.000B_{16}</td>
<td>XX_{16}</td>
<td>8 bit</td>
<td>Seven-Segment LED Display Control Register</td>
</tr>
<tr>
<td>7138.000C_{16}</td>
<td>FE_{16}</td>
<td>8 bit</td>
<td>FMB Channel 0 Data Discard Status Register</td>
</tr>
<tr>
<td>7138.000D_{16}</td>
<td>FE_{16}</td>
<td>8 bit</td>
<td>FMB Channel 1 Data Discard Status Register</td>
</tr>
<tr>
<td>7138.000E_{16}</td>
<td>XX_{16}</td>
<td>8 bit</td>
<td>reserved</td>
</tr>
<tr>
<td>7138.000F_{16}</td>
<td>FX_{16}</td>
<td>8 bit</td>
<td>LCA Identification Register</td>
</tr>
</tbody>
</table>
3.10 Front Panel

The Reset and Abort functions, the Hex display, the Rotary switch, and the LEDs are described on the following pages. The pinouts for the connectors shown in grey below are described in Section 2, Installation.

### Table 39: Front Panel Layout

<table>
<thead>
<tr>
<th>Device</th>
<th>Function</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>Reset</td>
<td>RESET</td>
</tr>
<tr>
<td>Switch</td>
<td>Abort</td>
<td>ABORT</td>
</tr>
<tr>
<td>HEX. Display</td>
<td>Diagnostic</td>
<td>DIAG</td>
</tr>
<tr>
<td>Rotary Switch</td>
<td>User defined</td>
<td>MODE</td>
</tr>
<tr>
<td>LED/LED</td>
<td>RUN/HALT VME Busmaster/SYSFAIL</td>
<td>RUN BM</td>
</tr>
<tr>
<td>LED/LED</td>
<td>Software programmable</td>
<td>01</td>
</tr>
<tr>
<td>MiniDIN Connector</td>
<td>Keyboard/Mouse</td>
<td>KBD</td>
</tr>
<tr>
<td>Serial Connector</td>
<td>Serial Interfaces</td>
<td>SERIAL A+B</td>
</tr>
<tr>
<td>SCSI Connector</td>
<td>SCSI Interface</td>
<td>SCSI</td>
</tr>
<tr>
<td>D-Sub Connector</td>
<td>Ethernet</td>
<td>ETHERNET</td>
</tr>
</tbody>
</table>
3.10.1  **RESET and ABORT Keys**

The front panel on the SPARC CPU-5V has two mechanical switches which directly influence the system.

3.10.1.1  **The RESET Key**

The **RESET** key enables the user to reset the whole board. If the board is VMEbus system controller (Slot-1 device), the SYSRESET signal of the VMEbus also becomes active with the **RESET** key. This resets the complete VMEbus system. The switch SW7-4 can be used to disable driving the VMEbus SYSRESET signal (see “VMEbus SYSRESET Enable/Disable” on page 81). With on-board switch SW7-1, it is possible to deactivate the **RESET** key. When SW7-1 is ON, the **RESET** key works and when SW7-1 is OFF, toggling the **RESET** key has no effect.

Please see also “VMEbus SYSRESET Enable/Disable” on page 81.

3.10.1.2  **The ABORT Key**

The **ABORT** key on the front panel can be used to generate a nonmaskable interrupt (level 15). The **ABORT** key function is controlled by switch SW7-2. When SW7-2 is ON, the key works and when SW7-2 is OFF, toggling the **ABORT** key has no effect. If the **ABORT** key produces a nonmaskable interrupt, the pending signal can be read in the Miscellaneous Control and Status Register 0 (MCSR0 register). The **ABORT** Interrupt Request Mapping Register (ABORT_IRQ_MAP) is used to map and enable an interrupt, generated by assertion of the ABKEY signal. Please see the *FGA-5000 Technical Reference Manual* for further information.
3.10.2 Front Panel Status LEDs

There are 4 single LEDs on the front panel.

- The RUN/RESET LED
- The VME BM LED (Bus Master)
- 2 STATUS LEDs

The RUN/RESET LED is either red, green or blinking. This LED is red when any reset signal on the board is active. This LED begins blinking when SB_SEL<4> is inactive for more than 0.5s in order to signal a hang up. In all other cases, this LED is green.

The BM LED reflects all VMEbus master activities on the CPU-5V. When the board accesses the VMEbus, the BM LED lights up green. The BM LED turns red when the CPU-5V is asserting SYSFAIL to the VMEbus.

There are 2 additional STATUS LEDs, which are freely programmable LEDs controlled by accessing registers in the LCA.
3.10.2.1 USER LED 1 Control Register

Physical Address 7138 0000_{16}

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>BLINK_FREQ</td>
<td>COLOUR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**COLOUR (RW)** These two bits are used to turn the first USER LED on or off, and to control the colour of the LED. The table below lists all possible values:

<table>
<thead>
<tr>
<th>COLOUR</th>
<th>Colour of the first USER LED (LED #0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00_2</td>
<td>USER LED is turned off</td>
</tr>
<tr>
<td>01_2</td>
<td>USER LED is turned on and shines green</td>
</tr>
<tr>
<td>10_2</td>
<td>USER LED is turned on and shines red</td>
</tr>
<tr>
<td>11_2</td>
<td>USER LED is turned on and shines yellow</td>
</tr>
</tbody>
</table>

**BLINK_FREQ (RW)** These two bits control the frequency at which the first USER LED is blinking. The table below lists all possible values and the corresponding blink frequency:

<table>
<thead>
<tr>
<th>BLINK_FREQ</th>
<th>( f_{\text{blink}} ) of the first USER LED (LED #0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00_2</td>
<td>USER LED is not blinking</td>
</tr>
<tr>
<td>01_2</td>
<td>USER LED is blinking at 1/2 Hz</td>
</tr>
<tr>
<td>10_2</td>
<td>USER LED is blinking at 1 Hz</td>
</tr>
<tr>
<td>11_2</td>
<td>USER LED is blinking at 2 Hz</td>
</tr>
</tbody>
</table>
3.10.2.2 USER LED 2 Control Register

Physical Address 7138 0001\textsubscript{16}

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>BLINK_FREQ</td>
<td>COLOUR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COLOUR (RW) These two bits are used to turn the second USER LED on or off, and to control the colour of the LED. The table below lists all possible values:

<table>
<thead>
<tr>
<th>COLOUR</th>
<th>Colour of the second USER LED (LED #1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00\textsubscript{2}</td>
<td>USER LED is turned off</td>
</tr>
<tr>
<td>01\textsubscript{2}</td>
<td>USER LED is turned on and shines green</td>
</tr>
<tr>
<td>10\textsubscript{2}</td>
<td>USER LED is turned on and shines red</td>
</tr>
<tr>
<td>11\textsubscript{2}</td>
<td>USER LED is turned on and shines yellow</td>
</tr>
</tbody>
</table>

BLINK\_FREQ (RW) These two bits control the frequency at which the second USER LED is blinking. The table below lists all possible values and the corresponding blink frequency:

<table>
<thead>
<tr>
<th>BLINK_FREQ</th>
<th>(f_{\text{blink}}) of the second USER LED (LED #1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00\textsubscript{2}</td>
<td>USER LED is not blinking</td>
</tr>
<tr>
<td>01\textsubscript{2}</td>
<td>USER LED is blinking at 1/2 Hz</td>
</tr>
<tr>
<td>10\textsubscript{2}</td>
<td>USER LED is blinking at 1 Hz</td>
</tr>
<tr>
<td>11\textsubscript{2}</td>
<td>USER LED is blinking at 2 Hz</td>
</tr>
</tbody>
</table>
3.10.3 Diagnostic LED (Hex Display)

A freely programmable LED display on the front panel provides diagnostic features. It can be accessed via the Seven Segment LED Display Control Register.

3.10.3.1 Seven Segment LED Display Control Register

Physical Address 7138 000B_{16}

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP</td>
<td>SEG_G</td>
<td>SEG_F</td>
<td>SEG_E</td>
<td>SEG_D</td>
<td>SEG_C</td>
<td>SEG_B</td>
<td>SEG_A</td>
</tr>
</tbody>
</table>

The following figure shows the hex display with the segments named in accordance to their bits in the Seven Segment LED Display Control Register. To switch a specific segment on, the corresponding bit must be set to one.

FIGURE 13. Segments of the Hex Display
3.10.4 Rotary Switch

The CPU-5V provides an additional rotary switch for user selectable settings. See the "Diagram of the CPU-5V (Top View)" on page 10 for the position of the rotary switch on the board. It is a hexadecimal rotary switch, decoded with 4 bits. The status of the rotary switch can be read in the Rotary Switch Status Register.

The table below shows the rotary switch settings and the corresponding values of the bits ROT[3..0], which you can read from the Rotary Switch Status Register.

### 3.10.4.1 Rotary Switch Status Register

<table>
<thead>
<tr>
<th>Physical Address 7138 000316</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

**ROTARY_SWITCH[3:0] (R)** These bits reflect the current state of the rotary switch. On the SPARC CPU-5V the rotary switch is connected in such a way to the LCA that the signals ROTARY_SWITCH[3:0] are inverted!

<table>
<thead>
<tr>
<th>Rotary Switch</th>
<th>ROTARY SWITCH [3:0]</th>
<th>Rotary Switch</th>
<th>ROTARY SWITCH [3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>016</td>
<td>11112</td>
<td>816</td>
<td>01112</td>
</tr>
<tr>
<td>116</td>
<td>11102</td>
<td>916</td>
<td>01102</td>
</tr>
<tr>
<td>216</td>
<td>11012</td>
<td>A16</td>
<td>01012</td>
</tr>
<tr>
<td>316</td>
<td>11002</td>
<td>B16</td>
<td>01002</td>
</tr>
<tr>
<td>416</td>
<td>10112</td>
<td>C16</td>
<td>00112</td>
</tr>
<tr>
<td>516</td>
<td>10102</td>
<td>D16</td>
<td>00102</td>
</tr>
<tr>
<td>616</td>
<td>10012</td>
<td>E16</td>
<td>00012</td>
</tr>
<tr>
<td>716</td>
<td>10002</td>
<td>F16</td>
<td>00002</td>
</tr>
</tbody>
</table>
3.11 Additional Registers

The following additional registers are provided on the CPU-5V to increase functionality.

3.11.1 FMB Channel 0 Data Discard Status Register

FMB channel 0 consists of an 8-stage FIFO and so does the FMB Channel 0 Data Discard Status Register. Read accesses to this register switch the internal read pointer one step ahead in the FIFO. Whenever your software needs to perform elementary functions as such, we recommend coordinating accesses to this register and the related FGA-5000 registers so that synchronisation of both FIFOs be not broken.

Physical Address 7138 000C\textsubscript{16}

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>MSG VALID</td>
</tr>
</tbody>
</table>

MSG\_VALID (R) The state of this bit indicates whether to discard the data in the FMB channel 0 of the SPARC FGA-5000. When the bit is cleared (0) then the data in the FMB channel must be discarded. In the case that this bit is set (1) the data in the FMB channel is valid.

3.11.2 FMB Channel 1 Data Discard Status Register

Physical Address 7138 000D\textsubscript{16}

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>MSG VALID</td>
</tr>
</tbody>
</table>

MSG\_VALID (R) The state of this bit indicates whether to discard the data in the FMB channel 1 of the SPARC FGA-5000. When the bit is cleared (0) then the data in the FMB channel must be discarded. In the case where this bit is set (1) the data in the FMB channel is valid.

A complete description of the FGA-5000 chip is found in the *FGA-5000 Technical Reference Manual*, available from FORCE COMPUTERS.
SECTION 4

4. Software

4.1 OpenBoot

This section describes the enhancements to the standard OpenBoot firmware that have been done for the SPARC CPU-5V. For a description of standard OpenBoot firmware features, please see the *OPEN BOOT PROM 2.0 MANUAL SET*.

Besides the commands already provided by the standard OpenBoot firmware, the OpenBoot firmware available on the SPARC CPU-5V includes further words for the following:

- accessing and controlling the VMEbus Interface,
- accessing and programming available flash memories,
- controlling the operating mode of the Watchdog Timer, and
- making use of the Diagnostics.

The following subsections describe these words in detail, and examples are given when it seems necessary to convey the usage of a particular or a group of words. In general, each word is described using the notation stated below:

```
name (stack-comment) description
```

The *name* field identifies the name of the word being described.

The stack parameters passed to and returned from a word are described by the *stack-comment* notation — enclosed in parentheses —, and show the effect of the word on the evaluation stack. The notation used is:

```
parameters before execution — parameters after execution
```

The parameters passed and returned to the word are separated by the “—”.

The *description* body describes the semantics of the word and conveys the purpose and effect of the particular word.

The OpenBoot ported to the SPARC CPU-5V is based upon the OpenBoot 2.15 obtained from Sun Microsystems.
4.2 VMEbus Interface

The VMEbus Interface on the SPARC CPU-5V consists of FORCE COMPUTERS’ SPARC FGA-5000 (VSI) chip. The FORCE Gate Array-5000 is a VMEbus to SBus interface chip.

4.2.1 Generic Information

The variables — which are declared as value — described below are used to retrieve generic information about the VMEbus interface:

vsi-va (— vaddr) returns the virtual base address vaddr of the registers included in the SPARC FGA-5000.

vsi-pa (— paddr) returns the physical base address paddr of the registers included in the SPARC FGA-5000.

vsi-sbus-slot# (— sbus-slot#) returns the number of the SBus slot sbus-slot# where the registers of the SPARC FGA-5000 are accessible.

vsi-offset (— offset) returns the offset within the particular SBus slot at which the registers, included in the SPARC FGA-5000, are accessible.

The base address of the SPARC FGA-5000, which is specified by the values vsi-sbus-slot# and vsi-offset, may be modified by the command described below:

vsi-base-addr! (offset sbus-slot# — ) sets the base address of the SPARC FGA-5000 according to the given SBus slot number sbus-slot# and the offset offset within the specified SBus slot. The values sbus-slot# and offset are stored in the appropriate variables vsi-sbus-slot# and vsi-offset. Furthermore, the command sets the variables vsi-pa and vsi-va according to the given parameters.

On the SPARC CPU-5V the SBus slots are utilized as stated in the table below.

<table>
<thead>
<tr>
<th>sbus-slot#</th>
<th>SSEL</th>
<th>Address Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>2000.0000_{16}...2FFFF.FFFF_{16}</td>
<td>SBus Slot #0 (AFX)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3000.0000_{16}...3FFFF.FFFF_{16}</td>
<td>SBus Slot #1 (reserved for VMEbus accesses through the SPARC FGA-5000)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4000.0000_{16}...4FFFF.FFFF_{16}</td>
<td>SBus Slot #2 (Sbus Card 1)</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5000.0000_{16}...5FFFF.FFFF_{16}</td>
<td>SBus Slot #3 (Sbus Card 2)</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>6000.0000_{16}...6FFFF.FFFF_{16}</td>
<td>SBus Slot #4 (Sbus Card 3)</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>7000.0000_{16}...7FFFF.FFFF_{16}</td>
<td>SBus Slot #5 (MACIO,SLAVIO, SPARC FGA-5000 Registers)</td>
</tr>
</tbody>
</table>
vsi-base-addr@ (— offset sbus-slot#) returns the base address of the SPARC FGA-5000 represented by the SBus slot number sbus-slot# and the offset offset within the specific SBus slot.

4.2.2 Register Addresses

The commands described below are used to obtain the virtual addresses of specific registers in the SPARC FGA-5000:

vsi-sbus-base (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Base Address Register.

vsi-id (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s Identification Register.

vsi-vme-range (range# — vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Address Decoding And Translation Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Address Decoding and Translation Registers.

vsi-vme-master-cap (range# — vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s VMEbus Master Capability Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 VMEbus Master Capability Registers.

vsi-vme-cap (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s VMEbus Capability Register.

vsi-sbus-ssel (range# — vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Slave Slot Select Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Slave Slot Select Registers.

vsi-sbus-master-cap (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Master Capability Register.

vsi-sbus-retry-time-ctrl (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Retry Time Control Register.

vsi-sbus-rerun-limit-ctrl (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Rerun Limit Control Register.
vsi-swpar(—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Write Posting Error Address Register.

vsi-vvpar(—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s VMEbus Write Posting Error Address Register.

vsi-slerr(—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Late Error Address Register.

vsi-slerr-irq-map(—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s SBus Late Error Interrupt Level Select and Enable Register.

vsi-iack-emu(level—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s IACK Cycle Emulation Register associated with the given level. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.

vsi-vme-base(—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s VMEbus Base Address Register.

vsi-sbus-range(range#—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s VMEbus Address Decoding and Translation Register identified by its range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Decoding and Translation Registers.

vsi-vme-ext(range#—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s VMEbus Address Extension Register identified by its range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Extension Registers.

vsi-reset-stat(—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s Reset Source Register.

vsi-intr-stat(—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s Interrupt Status Register.

vsi-irq-map(level—vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s VMEbus Interrupt Level Select and Enable Register associated with the given level. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.
vsi-mbox-irq-map (mailbox# — vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Mailbox Interrupt Level Select and Enable Register identified by its mailbox number mailbox#. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Interrupt Level Select and Enable Registers.

vsi-dma-irq-map (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's DMA Interrupt Level Select and Enable Register.

vsi-wpe-irq-map (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Write Posting Error Interrupt Level Select and Enable Register.

vsi-arb-irq-map (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Arbiter Timeout Interrupt Level Select and Enable Register.

vsi-wdt-irq-map (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Watchdog Timer Interrupt Level Select and Enable Register.

vsi-acfail-irq-map (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's ACFAIL Interrupt Level Select and Enable Register.

vsi-sysfail-irq-map0 (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's SYSFAIL Assert Interrupt Level Select and Enable Register.

vsi-sysfail-irq-map1 (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's SYSFAIL Negate Interrupt Level Select and Enable Register.

vsi-abort-irq-map (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Abort Interrupt Level Select and Enable Register.

vsi-arb-ctrl (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s Arbiter Control Register.

vsi-req-ctrl (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s Requester Control Register.

vsi-bus-ctrl (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000’s Bus Capture Control Register.

vsi-mbox (mailbox# — vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Mailbox Register identified by its mailbox number mailbox#. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vsi-mbox-stat (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's
Mailbox Status Register.

\( \text{vsi-sem( semaphore\# \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Semaphore Register identified by its semaphore number \( \text{semaphore}\# \). The value of \( \text{semaphore}\# \) may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.

\( \text{vsi-fmb-ctrl( \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Message Broadcast Control Register.

\( \text{vsi-fmb-irq-map( channel\# \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Message Broadcast Interrupt Level Select and Enable Register identified by its channel number \( \text{channel}\# \). The value of \( \text{channel}\# \) may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Interrupt Level Select and Enable Registers.

\( \text{vsi-fmb-addr( \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Message Broadcast Address Register.

\( \text{vsi-fmb-stat( channel\# \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Message Broadcast Status Register identified by its channel number \( \text{channel}\# \). The value of \( \text{channel}\# \) may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Status Registers.

\( \text{vsi-fmb-msg( channel\# \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Message Broadcast Register identified by its channel number \( \text{channel}\# \). The value of \( \text{channel}\# \) may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Registers.

\( \text{vsi-gcsr( \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Global Control and Status Register.

\( \text{vsi-wdt-ctrl( \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Watchdog Timer Control Register.

\( \text{vsi-wdt-restart( \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Watchdog Restart Register.

\( \text{vsi-mcsr0( \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Miscellaneous Control and Status Register 0.

\( \text{vsi-mcsrl( \rightarrow vaddr )} \) returns the virtual address \( vaddr \) of the SPARC FGA-5000's Miscellaneous Control and Status Register 1.
vsi-dma-ctrl (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's DMA Control Register.

vsi-dma-mode (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's DMA Mode Register.

vsi-dma-stat (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's DMA Status Register.

vsi-dma-src (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's DMA Source Address Register.

vsi-dma-dest (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's DMA Destination Address Register.

vsi-dma-cap (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's DMA Capability and Transfer Count Register.

vsi-ibox-irq-map (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Interrupt Box Interrupt Level Select and Enable Register.

vsi-ibox-ctrl (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Interrupt Box Control Register.

vsi-ibox-addr (— vaddr) returns the virtual address vaddr of the SPARC FGA-5000's Interrupt Box Address Register.

The following commands are available to get the virtual addresses of the System Configuration Registers.

sysconfig-va (— vaddr) returns the virtual base address vaddr of the System Configuration Registers.

led1-ctrl (— vaddr) returns the virtual address vaddr of the First User LED Control Register.

led2-ctrl (— vaddr) returns the virtual address vaddr of the Second User LED Control Register.

flash-ctrl1 (— vaddr) returns the virtual address vaddr of the Flash Memory Control Register 1.

rotary-switch-stat (— vaddr) returns the virtual address vaddr of the Rotary Switch Status Register.
boot-rom-size-ctrl (— vaddr) returns the virtual address vaddr of the Boot ROM Size Control Register.

flash-ctrl2 (— vaddr) returns the virtual address vaddr of the Flash Memory Control Register 2.

flash-vpp-ctrl (— vaddr) returns the virtual address vaddr of the Flash Memory Programming Voltage Control Register.

led-display-ctrl (— vaddr) returns the virtual address vaddr of the LED Display Control Register.

fmb-0-data-discard (— vaddr) returns the virtual address vaddr of the FMB Channel 0 Data Discard Status Register.

fmb-1-data-discard (— vaddr) returns the virtual address vaddr of the FMB Channel 1 Data Discard Status Register.

lca-id (— vaddr) returns the virtual address vaddr of the LCA ID Register.

4.2.3 Register Accesses

The commands described below are used to read data from and to store data in specific registers of the SPARC FGA-5000:

vsi-sbus-base@ (— long) returns the contents — a 32-bit data — of the SBus Base Address Register.

vsi-sbus-base! (long —) stores the 32-bit data long in the SBus Base Address Register.

vsi-id@ (— id-code) returns the contents — the 32-bit data id-code — of the Identification Register.

vsi-vme-range@ (range# — long) returns the contents — a 32-bit data — of the SBus Address Decoding And Translation Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Address Decoding and Translation Registers.

vsi-vme-range! (long range# —) stores the 32-bit value long in the SBus Address Decoding And Translation Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Address Decoding and Translation Registers.
vsi-vme-master-cap@ (range# — byte) returns the contents — an 8-bit data — of the VMEbus Master Capability Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 VMEbus Master Capability Registers.

vsi-vme-master-cap! (byte range# — ) stores the 8-bit data byte in the VMEbus Master Capability Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 VMEbus Master Capability Registers.

vsi-vme-cap@ (— byte) returns the contents — an 8-bit data — of the VMEbus Capability Register.

vsi-vme-cap! (byte — ) stores the 8-bit data byte in the VMEbus Master Capability Register.

vsi-sbus-ssel@ (range# — byte) returns the contents — an 8-bit data — of the SBus Slave Slot Select Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Slave Slot Select Registers.

vsi-sbus-ssel! (byte range# — ) stores the 8-bit data byte in the SBus Slave Slot Select Register identified by its register number range#. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Slave Slot Select Registers.

vsi-sbus-cap@ (— byte) returns the contents — an 8-bit data — of the SPARC FGA-5000’s SBus Capability Register.

vsi-sbus-cap! (byte — ) stores the 8-bit data byte in the SPARC FGA-5000’s SBus Capability Register.

vsi-sbus-retry-time-ctrl@ (— byte) returns the contents — an 8-bit data — of the SPARC FGA-5000’s SBus Retry Time Control Register.

vsi-sbus-retry-time-ctrl! (byte — ) stores the 8-bit data byte in the SPARC FGA-5000’s SBus Retry Time Control Register.

vsi-sbus-rerun-limit-ctrl@ (— word) returns the contents — a 16-bit data — of the SPARC FGA-5000’s SBus Rerun Limit Control Register.

vsi-sbus-rerun-limit-ctrl! (word — ) store the 16-bit data word in the SPARC FGA-5000’s SBus Rerun Limit Control Register.
vsiswpar@(-long) returns the contents — a 32-bit data — of the SBus Write Posting Error Address Register.

vsivwpar@(-long) returns the contents — a 32-bit data — of the VMEbus Write Posting Error Address Register.

vslerr@(-long) returns the contents — a 32-bit data — of the SBus Late Error Address Register.

vslerr-irq-map@(-byte) returns the contents — an 8-bit data — of the Late Error Interrupt Level Select and Enable Register.

vslerr-irq-map!(byte-) stores the 8-bit data byte in the Late Error Interrupt Level Select and Enable Register.

vsiack-emu@(level-byte) returns the contents — an 8-bit data — of the IACK Cycle Emulation Register associated with the given level. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.

vserme-base@(-byte) returns the contents — an 8-bit data — of the VMEbus Base Address Register.

vserme-base!(byte-) stores the 8-bit data byte in the VMEbus Base Address Register.

vsisbus-range@(range#-long) returns the contents — a 32-bit data — of the VMEbus Address Decoding and Translation Register identified by its range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Decoding and Translation Registers.

vsisbus-range!(long range#-) stores the 32-bit data long in the VMEbus Address Decoding and Translation Register identified by its range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Decoding and Translation Registers.

vsive-ext@(range#-byte) returns the contents — an 8-bit data — of the VMEbus Address Extension Register identified by its range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Extension Registers.
vsi-vme-ext!(byte range# — ) stores the 8-bit data byte in the VMEbus Address Extension Register identified by its range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Extension Registers.

vsi-reset-stat@ (— byte ) returns the contents — an 8-bit data — of the Reset Source Register.

gsi-intr-stat@ (— long ) returns the contents — a 32-bit data — of the Interrupt Status Register.

gsi-intr-stat! (long — ) stores the 32-bit data long in the Interrupt Status Register.

.vsi-intr-stat (— ) displays the actual contents of the Interrupt Status Register. The contents of the register is displayed as shown below:

```
ok .vsi-intr-stat
VME-IRQ1: 0 VME-IRQ2: 0 VME-IRQ3: 0 VME-IRQ4: 0 VME-IRQ5: 0
VME-IRQ6: 0 VME-IRQ7: 0 VME-IACK: 0 FMBl : 0 FMB0 : 0
IBOX : 0 LERR : 0 WDOG : 0 DMATERM : 0 VWPERR : 0
SWPERR : 0 MAILBOX : 0 ARbstOUT : 0 ABORT : 0 SYSFAIL+: 0
SYSFAIL- : 0 ACFAIL : 0
ok
```

When an interrupt is pending the command displays the one (1); otherwise it displays the zero (0) to indicate that the interrupt is not pending.

**Note!** The state of the entry SYSFAIL- reports the occurrence of a *negative* edge of the VMEbus SYSFAIL* signal which indicates that the SYSFAIL* signal has been *asserted*. The state of the entry SYSFAIL+ reports the occurrence of a *positive* edge of the VMEbus SYSFAIL* signal which indicates that the SYSFAIL* signal has been *negated*.

vsi-irq-map@ (level— byte ) returns the contents — an 8-bit data — of the VMEbus Interrupt Level Select and Enable Register associated with the given level. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.

Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.

vsi-irq-map! (byte level — ) stores the 8-bit data byte in the VMEbus Interrupt Level Select and Enable Register associated with the given level. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.

Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.
vsi-mbox-irq-map@ (mailbox# — byte) returns the contents — an 8-bit data — of the Mailbox Interrupt Level Select and Enable Register identified by its mailbox number mailbox#. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Interrupt Level Select and Enable Registers.

vsi-mbox-irq-map! (byte mailbox# — ) stores the 8-bit data byte in the Mailbox Interrupt Level Select and Enable Register identified by its mailbox number mailbox#. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Interrupt Level Select and Enable Registers.

vsi-dma-irq-map@ ( — byte) returns the contents — an 8-bit data — of the DMA Interrupt Level Select and Enable Register.

vsi-dma-irq-map! (byte — ) stores the 8-bit data in the DMA Interrupt Level Select and Enable Register.

vsi-wpe-irq-map@ ( — byte) returns the contents — an 8-bit data — of the Write Posting Error Interrupt Level Select and Enable Register.

vsi-wpe-irq-map! (byte — ) stores the 8-bit data byte in the Write Posting Error Interrupt Level Select and Enable Register.

vsi-arb-irq-map@ ( — byte) returns the contents — an 8-bit data — of the Arbiter Timeout Interrupt Level Select and Enable Register.

vsi-arb-irq-map! (byte — ) stores the 8-bit data byte in the Arbiter Timeout Interrupt Level Select and Enable Register.

vsi-wdt-irq-map@ ( — byte) returns the contents — an 8-bit data — of the Watchdog Timer Interrupt Level Select and Enable Register.

vsi-wdt-irq-map! (byte — ) stores the 8-bit data byte in the Watchdog Timer Interrupt Level Select and Enable Register.

vsi-acfail-irq-map@ ( — byte) returns the contents — an 8-bit data — of the ACFAIL Interrupt Level Select and Enable Register.

vsi-acfail-irq-map! (byte — ) stores the 8-bit data byte in the ACFAIL Interrupt Level Select and Enable Register.

vsi-sysfail-irq-map0@ ( — byte) returns the contents — an 8-bit data — of the SYSFAIL Assert Interrupt Level Select and Enable Register.
vsi-sysfail-irq-map! (byte —) stores the 8-bit data byte in the SYSFAIL Assert Interrupt Level Select and Enable Register.

vsi-sysfail-irq-map! (byte —) stores the 8-bit data byte in the SYSFAIL Negate Interrupt Level Select and Enable Register.

vsi-sysfail-irq-map! (byte —) stores the 8-bit data byte in the SYSFAIL Negate Interrupt Level Select and Enable Register.

vsi-arb-ctrl@ (— byte ) returns the contents — an 8-bit data — of the Arbiter Control Register.

vsi-arb-ctrl! (byte —) stores the 8-bit data long in the Arbiter Control Register.

vsi-req-ctrl@ (— byte ) returns the contents — an 8-bit data — of the Requester Control Register.

vsi-req-ctrl! (byte —) stores the 8-bit data byte in the Requester Control Register.

vsi-bus-ctrl@ (— byte ) returns the contents — an 8-bit data — of the Bus Capture Control Register.

vsi-bus-ctrl! (byte —) stores the 8-bit data byte in the Bus Capture Control Register.

vsi-mbox@ (mailbox#— byte ) returns the contents — an 8-bit data — of the Mailbox Register identified by its mailbox number mailbox#. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vsi-mbox! (byte mailbox#— ) stores the 8-bit data byte in the Mailbox Register identified by its mailbox number mailbox#. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vsi-mbox-stat@ (— word ) returns the contents — a 16-bit data — of the Mailbox Status Register.

vsi-mbox-stat! (word —) stores the 16-bit data long in the Mailbox Status Register.

vsi-sem@ (semaphore#— byte ) returns the contents — an 8-bit data — of the Semaphore Register identified by its semaphore number semaphore#. The value of semaphore# may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.
vsi-sem! (byte semaphore# — ) stores the 8-bit data byte in the Semaphore Register identified by its semaphore number semaphore#. The value of semaphore# may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.

vsi-fmb-ctrl@ (— byte) returns the contents — an 8-bit data — of the Message Broadcast Control Register.

vsi-fmb-ctrl! (byte — ) stores the 8-bit data byte in the Message Broadcast Control Register.

vsi-fmb-irq-map@ (channel# — byte ) returns the contents — an 8-bit data — of the Message Broadcast Interrupt Level Select and Enable Register identified by its channel number channel#. The value of channel# may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Interrupt Level Select and Enable Registers.

vsi-fmb-irq-map! (byte channel# — ) stores the 8-bit data byte in the Message Broadcast Interrupt Level Select and Enable Register identified by its channel number channel#. The value of channel# may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Interrupt Level Select and Enable Registers.

vsi-fmb-addr@ (— byte ) returns the contents — an 8-bit data — of the Message Broadcast Address Register.

vsi-fmb-addr! (byte — ) stores the 8-bit data byte in the Message Broadcast Address Register.

vsi-fmb-stat@ (channel# — byte ) returns the contents — an 8-bit data — of the Message Broadcast Status Register identified by its channel number channel#. The value of channel# may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Status Registers.

vsi-fmb-stat! (byte channel# — ) stores the 8-bit data byte in the Message Broadcast Status Register identified by its channel number channel#. The value of channel# may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Status Registers.

vsi-fmb-msg@ (channel# — long true | false ) returns the contents — a 32-bit data — of the Message Broadcast Register identified by its channel number channel#. The value of channel# may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Registers.
vsi-gcsr@ (— byte) returns the contents — an 8-bit data — of the Global Control and Status Register.

vsi-gcsr! (byte —) stores the 8-bit data byte in the Global Control and Status Register.

vsi-mcsr0@ (— byte) returns the contents — an 8-bit data — of the Miscellaneous Control and Status Register 0.

vsi-mcsr0! (byte —) stores the 8-bit data byte in the Miscellaneous Control and Status Register 0.

vsi-mcsr1@ (— byte) returns the contents — an 8-bit data — of the Miscellaneous Control and Status Register 1.

vsi-mcsr1! (byte —) stores the 8-bit data byte in the Miscellaneous Control and Status Register 1.

vsi-wdt-ctrl@ (— byte) returns the contents — an 8-bit data — of the Watchdog Timer Control Register.

vsi-wdt-ctrl! (byte —) stores the 8-bit data byte in the Watchdog Timer Control Register.

vsi-wdt-restart@ (— byte) returns the contents — an 8-bit data — of the Watchdog Restart Register.

vsi-wdt-restart! (byte —) stores the 8-bit data byte in the Watchdog Restart Register.

vsi-dma-ctrl@ (— word) returns the contents — a 16-bit data — of the DMA Control Register.

vsi-dma-ctrl! (word —) stores the 16-bit data word in the DMA Control Register.

vsi-dma-mode@ (— byte) returns the contents — an 8-bit data — of the DMA Mode Register.

vsi-dma-mode! (byte —) stores the 8-bit data byte in the DMA Mode Register.

vsi-dma-stat@ (— byte) returns the contents — an 8-bit data — of the DMA Status Register.

vsi-dma-stat! (byte —) stores the 8-bit data byte in the DMA Status Register.

vsi-dma-src@ (— long) returns the contents — a 32-bit data — of the DMA Source Address Register.
vsi-dma-src!(long — ) stores the 32-bit data long in the DMA Source Address Register.

vsi-dma-dest@(-long) returns the contents — a 32-bit data — of the DMA Destination Address Register.

vsi-dma-dest!(long — ) stores the 32-bit data long in the DMA Destination Address Register.

vsi-dma-cap@(-long) returns the contents — a 32-bit data — of the DMA Capability and Transfer Count Register.

vsi-dma-cap!(long — ) stores the 32-bit data long in the DMA Capability and Transfer Count Register.

vsi-ibox-irq-map@(-byte) returns the contents — an 8-bit data — of the Interrupt Box Interrupt Level Select and Enable Register.

vsi-ibox-irq-map!(byte — ) stores the 32-bit data long in the Interrupt Box Interrupt Level Select and Enable Register.

vsi-ibox-ctl@(-word) returns the contents — a 16-bit data — of the Interrupt Box Control Register.

vsi-ibox-ctl!(word — ) stores the 16-bit data word in the Interrupt Box Control Register.

vsi-ibox-addr@(-word) returns the contents — a 16-bit data — of the Interrupt Box Address Register.

vsi-ibox-addr!(word — ) stores the 16-bit data word in the Interrupt Box Address Register.

The following commands are available to read data from and store data in the System Configuration Registers.

led1-ctl@(-byte) returns the contents — an 8-bit data — of the First User LED Control Register.

led1-ctl!(byte — ) stores the 8-bit data byte in the First User LED Control Register.

led2-ctl@(-byte) returns the contents — an 8-bit data — of the Second User LED Control Register.

led2-ctl!(byte — ) stores the 8-bit data byte in the Second User LED Control Register.
flash-ctrl1@ (— byte ) returns the contents — an 8-bit data — of the Flash Memory Control Register 1.

flash-ctrl1! ( byte — ) stores the 8-bit data byte in the Flash Memory Control Register 1.

rotary-switch-stat@ (— byte ) returns the contents — an 8-bit data — of the Rotary Switch Status Register.

boot-rom-size-ctr1@ (— byte ) returns the contents — an 8-bit data — of the Boot ROM Size Control Register.

boot-rom-size-ctr1! ( byte — ) stores the 8-bit data byte in the Boot ROM Size Control Register.

flash-ctr12@ (— byte ) returns the contents — an 8-bit data — of the Flash Memory Control Register 2.

flash-ctr12! ( byte — ) stores the 8-bit data byte in the Flash Memory Control Register 2.

flash-vpp-ctr1@ (— byte ) returns the contents — an 8-bit data — of the Flash Memory Programming Voltage Control Register.

flash-vpp-ctr1! ( byte — ) stores the 8-bit data byte in the Flash Memory Programming Voltage Control Register.

led-display-ctr1@ ( byte — ) returns the contents — an 8-bit data — of the LED Display Control/Status Register. Because the LED Display Control Register is only writable, the command returns the contents of the LED Display Control Shadow Register.

led-display-ctr1! (— byte ) stores the 8-bit data byte in the LED Display Control/Status Register. Because the LED Display Control Register is only writable, the command stores the given data in the LED Display Control Shadow Register, too.

fmb-0-data-discard@ (— byte ) returns the contents — an 8-bit data — of the FMB Channel 0 Data Discard Status Register.

fmb-1-data-discard@ (— byte ) returns the contents — an 8-bit data — of the FMB Channel 1 Data Discard Status Register.

lca-id@ (— byte ) returns the contents — an 8-bit data — of the LCA ID Register.
4.2.4 VMEbus Interrupt Handler

vme-intr-pending? ( level — true \ false ) checks whether an interrupt is pending on a given interrupt request level and returns a flag. When an interrupt is pending the flag is true; otherwise it is false. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command. The command verifies the state of the interrupt pending bit in the Interrupt Status register associated with the given level. When the corresponding status bit is set then no VMEbus interrupt is pending and the command returns false. Otherwise — the status bit is cleared — the value true is returned.

vme-iack@ ( level — vector ) initiates an interrupt acknowledge cycle at the given VMEbus interrupt request level and returns the obtained 8-bit vector. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Typically, the vector returned is within the range 0 through 255, but when no interrupt is pending, and therefore no interrupt has to be acknowledged, the value -1 is returned. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.
vme-intr-ena (mapping level — ) enables the interrupt to be generated upon the receipt of a VMEbus interrupt at level. The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of mapping may be one of the values in the range one through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table below lists all allowed mappings.

The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of mapping and level are considered. When level is zero then the command treats it as if the value “one” has been passed to the command.

<table>
<thead>
<tr>
<th>mapping</th>
<th>Constant</th>
<th>Interrupt generated by SPARC FGA-5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>vsi-nmi</td>
<td>INT (connected with nonmaskable interrupt)</td>
</tr>
<tr>
<td>1</td>
<td>vsi-sbus-irq-1</td>
<td>SINT1 (connected with SBus IRQ1)</td>
</tr>
<tr>
<td>2</td>
<td>vsi-sbus-irq-2</td>
<td>SINT2 (connected with SBus IRQ2)</td>
</tr>
<tr>
<td>3</td>
<td>vsi-sbus-irq-3</td>
<td>SINT3 (connected with SBus IRQ3)</td>
</tr>
<tr>
<td>4</td>
<td>vsi-sbus-irq-4</td>
<td>SINT4 (connected with SBus IRQ4)</td>
</tr>
<tr>
<td>5</td>
<td>vsi-sbus-irq-5</td>
<td>SINT5 (connected with SBus IRQ5)</td>
</tr>
<tr>
<td>6</td>
<td>vsi-sbus-irq-6</td>
<td>SINT6 (connected with SBus IRQ6)</td>
</tr>
<tr>
<td>7</td>
<td>vsi-sbus-irq-7</td>
<td>SINT7 (connected with SBus IRQ7)</td>
</tr>
</tbody>
</table>

The words listed in the second column of the table may be used to specify a valid interrupt mapping.

Table 40: Interrupt Mapping.

vme-intr-dis (level — ) disables the interrupt to be generated when the specified VMEbus interrupt request at level is asserted. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.

vme-irq-ena (level — ) enables the interrupt to be generated upon the receipt of a VMEbus interrupt at level. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.
vme-irq-dis (level — ) disables the interrupt to be generated upon the receipt of a VMEbus interrupt at level. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.

vme-irq-map@ (level — mapping) returns the interrupt asserted by the SPARC FGA-5000 when the VMEbus interrupt request level is asserted. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.

vme-irq-map! (mapping level — ) defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of mapping may be one of the values in the range one through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table “Interrupt Mapping.” on page 109 lists all allowed mappings. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of mapping and level are considered. When level is zero then the command treats it as if the value “one” has been passed to the command.

install-vme-intr-handler (mapping level — ) installs the interrupt service routine dealing with the given VMEbus interrupt level. The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table “Interrupt Mapping.” on page 109 lists all allowed mappings. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. The address of the interrupt service routine currently in effect is preserved. Only the least significant three bits of mapping and level are considered. When level is zero then the command treats it as if the value “one” has been passed to the command.

uninstall-vme-intr-handler (level — ) removes the interrupt service routine dealing with the given VMEbus interrupt level and installs the old interrupt service routine. The value of level may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of level are considered and when level is zero then the command treats it as if the value “one” has been passed to the command.

vme-vectors (— ) displays the VMEbus interrupt vectors received during the last interrupt acknowledge cycle.
OpenBoot maintains seven variables called vme-intr{1|2|3|4|5|6|7}-vec-
tor which are modified by the VMEbus interrupt handlers. In general, the interrupt handlers store the vector obtained during an interrupt acknowledge cycle in the appropriate variable.
4.2.5 VMEbus Arbiter

The commands listed below are available to control the arbiter:

vme-slotl-ena (—) enables the board to act as the system controller. In particular the command calls vme-slotl! — passing the value true to it — to enable the system controller function.

vme-slotl-dis (—) disables the board to act as the system controller. In particular the command calls vme-slotl! — passing the value false to it — to disable the system controller function.

vme-slotl! (true | false —) enables or disables the board’s function to operate as the system controller. When the value true is passed to the command the board acts as the system controller. Otherwise — the value false is passed to the command — the system controller function is disabled.

vme-arb-mode@ (— mode ) returns the mode the arbiter is currently operating in. The value of mode may range from zero to three. Each value specifies a particular mode: the values zero and three indicate that the arbiter is operating in the priority mode; the value one specifies the round-robin mode; and the value two specifies the prioritized-round-robin mode.

Three constants are available to specify one of the three bus arbitration modes: pri — prioritized — (310), rrs — round robin select— (110), prr — prioritized round robin — (610).

vme-arb-mode! ( mode — ) selects the arbiter mode specified by mode. The value of mode may range from zero to three. Each value specifies a particular mode: the values zero and three indicate that the arbiter operates in the priority mode; the value one specifies the round-robin mode; and the value two specifies the prioritized-round-robin mode.

vme-arb-irq-map! ( mapping — ) selects the interrupt to be generated by the arbiter when the arbitration timeout expired. The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table “Interrupt Mapping.” on page 109 lists all allowed mappings.

vme-arb-irq-ena (—) enables the interrupt to be generated by the arbiter when the arbitration timeout expired. In particular the command calls vme-arb-irq! — passing the value true to it — to enable the interrupt.

vme-arb-irq-dis (—) disables the interrupt to be generated by the arbiter when the arbitration timeout expired. In particular the command calls vme-arb-irq! — passing the value false to it — to disable the interrupt.
vme-arb-irq! (true | false — ) enables or disables the interrupt to be generated by the arbiter when the arbitration timeout expired. When the value true is passed to the command the interrupt is enabled. Otherwise — the value false is passed to the command — the interrupt is disabled.

.vsi-arb-ctrl(—) displays the current contents of the VMEbus Arbiter Control Register.

4.2.6 VMEbus Requester

The commands listed below are available to control the requester and to obtain some information about the requester's operational state:

vme-bus-request-level@ (— level ) returns the VMEbus request level in use when the VMEbus interface tries to gain the ownership of the VMEbus. The value of level may be one of the values in the range one through three. Each value specifies one of the four VMEbus request levels.

vme-bus-request-level! (level —) selects the bus-request level to be used when the VMEbus is being accessed. The value of level may be one of the values in the range one through three. Each value specifies one of the four VMEbus request levels.

vme-bus-request-mode@ (— mode ) returns the VMEbus request mode in use when the VMEbus interface tries to gain the ownership of the VMEbus.

vme-bus-request-mode! (mode —) selects the bus-request mode to be used when the VMEbus is being accessed. Two constants are available to specify one of the two request modes: fair (010) and unfair (110).

vme-bus-release-mode@ (— mode ) returns the VMEbus release mode in use when the VMEbus interface has gained the ownership of the VMEbus.

vme-bus-release-mode! (mode —) selects the release mode to be used when the VMEbus interface has gained the ownership of the VMEbus. Four constants are available to specify one of the four release modes: ror - release on request — (310), roc - release on bus clear — (510), rat - release after timeout — (610), and rwd - release when done — (710).

Because the SPARC FGA-5000 allows to consider more than one bus release mode simultaneously — however, the combination of the release modes should be reasonable — the following two examples show how to use the available constants and command to specify the release mode:
In this example the VMEbus will be released either when another master requests the VMEbus, or after a fixed timeout expired. In the example below the VMEbus is released either when the BBSY* signal is negated, or after a fixed timeout expired.

vme-early-release! (true | false —) allows or prevents the requester from releasing the VMEbus early. When the value true is passed to the command the requester releases the VMEbus before the cycle has been terminated completely. Otherwise — the value false is passed to the command — the requester releases the VMEbus only when the cycle has been terminated completely.

vme-bbsy-filter! (true | false —) enables or disables the BBSY* glitch filter. When the value true is passed to the command the BBSY* glitch filter is enabled. Otherwise — the value false is passed to the command — the BBSY* glitch filter is disabled.

.vsi-req-ctrl (—) displays the current contents of the VMEbus Requester Control Register.

vme-bus-capture! (true | false —) enables or disables the bus-capture-and-hold capability of the SPARC FGA-5000. If the value true is passed to the command the VMEbus Interface starts to capture the bus and when it gains the ownership of the bus it holds the as long as the bus is released. The bus is released when the command is called and the value false is passed to it.

vme-bus-captured? (— true | false ) determines whether the VMEbus interface gains the ownership of the bus. The value true is returned when the VMEbus interface gains the ownership of the VMEbus. Otherwise the value false is returned to indicated that the VMEbus interface has not gained the ownership of the bus.

In general this command is called immediately after a capture-and-hold cycle has been initiated as shown in the example below:

ok true vme-bus-capture!
ok begin vme-bus-captured? until
ok ...
ok false vme-bus-capture!
ok
4.2.7 VMEbus Status Signals

The commands listed below are available to access and control the VMEbus status signals.

vme-sysfail-set (-) asserts (sets) the VMEbus SYSFAIL* signal.

vme-sysfail-clear (-) negates (clears) the VMEbus SYSFAIL* signal.

vme-sysfail? (- true | false ) determines the state of the VMEbus SYSFAIL* signal and returns a flag set according to the signal’s state. When the SYSFAIL* signal is asserted the flag returned is true; otherwise its value is false.

vme-sysfail-assert-irq-map! (mapping — ) selects the interrupt to be generated when the VMEbus SYSFAIL* signal is asserted. The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the SYSFAIL* signal is asserted. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table “Interrupt Mapping.” on page 109 lists all allowed mappings.

vme-sysfail-assert-irq-ena (-) allows the VMEbus interface to generate an interrupt upon the assertion of the VMEbus SYSFAIL* signal.

vme-sysfail-assert-irq-dis (-) disables the interrupt to be generated upon the assertion of the VMEbus SYSFAIL* signal.

vme-sysfail-assert-ip? (- true | false ) checks whether an interrupt is pending due to the assertion of the VMEbus SYSFAIL* signal and returns a flag set according to the appropriate interrupt pending flag. The flag is true when the interrupt is pending; otherwise its value is false.

vme-sysfail-assert-ip-clear (-) clears a pending interrupt generated by the assertion of the VMEbus SYSFAIL* signal. This command clears on the corresponding interrupt pending bit in the Interrupt Status Register of the SPARC FGA-5000.

vme-sysfail-negate-irq-map! (mapping — ) selects the interrupt to be generated when the VMEbus SYSFAIL* signal is negated. The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the SYSFAIL* signal is negated. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table “Interrupt Mapping.” on page 109 lists all allowed mappings.

vme-sysfail-negate-irq-ena (-) allows the VMEbus interface to generate an interrupt upon the negation of the VMEbus SYSFAIL* signal.
vme-sysfail-negate-irq-dis (-) disables the interrupt to be generated upon the negation of the VMEbus SYSFAIL* signal.

vme-sysfail-negate-ip? (— true | false) checks whether an interrupt is pending due to the negation of the VMEbus SYSFAIL* signal and returns a flag set according to the appropriate interrupt pending flag. The flag is true when the interrupt is pending; otherwise its value is false.

vme-sysfail-negate-ip-clear (—) clears a pending interrupt generated by the negation of the VMEbus SYSFAIL* signal. This command clears on the corresponding interrupt pending bit in the Interrupt Status Register of the SPARC FGA-5000.

vme-acfail? (— true | false) determines the state of the VMEbus ACFAIL* signal and returns a flag set according to the signal's state. When the ACFAIL* signal is asserted the flag returned is true; otherwise it is false.

vme-acfail-assert-irq-map! (mapping —) selects the interrupt to be generated when the VMEbus ACFAIL* signal is asserted. The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the ACFAIL* signal is asserted. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping," on page 109 lists all allowed mappings.

vme-acfail-assert-irq-ena (—) allows the VMEbus interface to generate an interrupt upon the assertion of the VMEbus ACFAIL* signal.

vme-acfail-assert-irq-dis (—) disables the interrupt to be generated upon the assertion of the VMEbus ACFAIL* signal.

vme-acfail-assert-ip? (— true | false) checks whether an interrupt is pending due to the assertion of the VMEbus ACFAIL* signal and returns a flag set according to the appropriate interrupt pending flag. The flag is true when the interrupt is pending; otherwise its value is false.

vme-acfail-assert-ip-clear (—) clears a pending interrupt generated by the assertion of the VMEbus ACFAIL* signal. This command clears on the corresponding interrupt pending bit in the Interrupt Status Register of the SPARC FGA-5000.
4.2.8 VMEbus Master Interface

The SPARC FGA-5000 provides 16 sets of registers to control any VMEbus master operation. Each set may be used to address a certain address range within the VMEbus' address space. A register set is identified by an unique number, the range number (range#), in the range zero through 15.

When the VMEbus is being accessed the part of the SPARC FGA-5000 connected with the SBus is considered as the SBus slave device, whereas the part of the SPARC FGA-5000 that is connected with the VMEbus is operating as VMEbus master. This fact is reflected in the names of the commands available to control the VMEbus master interface.

The commands listed and described in the following are available to initialize and control the VMEbus master interface:

- `#vme-ranges` returns the number of available register sets which are used to control accesses to the VMEbus.

- `vme-master-ena` enables the address decoding associated with the range number to access the VMEbus. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

- `vme-master-dis` enables the address decoding associated with the range number to access the VMEbus. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

- `vme-master-wp-ena` enables write posting within the VMEbus address range associated with the range number. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

- `vme-slave-wp-dis` disables write posting within the VMEbus address range associated with the range number. The value of range# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

- `vme-supervisor!` selects the mode in which the VMEbus is being accessed. When the value true is passed to the command, the VMEbus is accessed in the privileged mode. Otherwise, the value false is passed to the command — the VMEbus is accessed in the non-privileged mode.

The mode selected with this command applies to all ranges used to access the VMEbus.

- `vme-master-cap@` returns the address- and data capabilities associated with the range number which are used when
the VMEbus is accessed. The value of \textit{range}# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation. The value of \textit{data-capability} and \textit{address-capability} may be one of the values listed in the table below.

\texttt{vme-master-cap! (data-capability address-capability range# —)} defines the address- and data capabilities associated with the range number \textit{range}# which are used when the VMEbus is accessed. The value of \textit{range}# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation. The value of \textit{data-capability} and \textit{address-capability} may be one of the values listed in the table below:

\begin{table}[h]
\begin{tabular}{|c|c|c|}
\hline
value & data-capability & address-capability \\
\hline
000\textsubscript{2} & cap-d8 & cap-a16 \\
\hline
001\textsubscript{2} & cap-d16 & cap-a24 \\
\hline
010\textsubscript{2} & cap-d32 & cap-a32 \\
\hline
011\textsubscript{2} & cap-blit & reserved \\
\hline
100\textsubscript{2} & cap-mblt & cap-a64 \\
\hline
101\textsubscript{2} & reserved & reserved \\
\hline
110\textsubscript{2} & reserved & reserved \\
\hline
111\textsubscript{2} & reserved & reserved \\
\hline
\end{tabular}
\end{table}

\texttt{sbus-slot-sel@ (range# — sbus-slot#)} returns the number of the SBus slot \textit{sbus-slot#} that is associated with the range identified by \textit{range}#. The value of \textit{range}# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

\texttt{sbus-slot-sel! (sbus-slot# range# —)} sets the number of the SBus slot \textit{sbus-slot#} that is associated with the range identified by \textit{range}#. The value of \textit{range}# may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

\texttt{sbus-slot#>ssel# (sbus-slot# — ssel#)} converts the logical SBus slot number \textit{sbus-slot#} to the corresponding SBus slave select number \textit{ssel#}.

\texttt{ssel#>sbus-slot# (ssel# — sbus-slot#)} converts the SBus slave select number \textit{ssel#} to the corresponding logical SBus slot number \textit{sbus-slot#}.
The `/sbus-range (range# — size)` command returns the size of the range associated with the range number `range#`. The value of `range#` may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

The `sbus-slave-range (range# — offset sbus-slot# size)` command returns the SBus slave parameters associated with the range identified by `range#`. The value of `range#` may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation. The parameters returned by the command specify the SBus address range to be accessed to reach the VMEbus. The address range is represented by the triple `offset`, `sbus-slot#`, and `size`.

The `sbus-slave-range! (offset sbus-slot# size range# — )` command sets the SBus slave parameters associated with the range identified by `range#`. The value of `range#` may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation. The parameters passed to the command specify the SBus address range to be accessed to reach the VMEbus. The address range is represented by the triple `offset`, `sbus-slot#`, and `size`.

The `vme-master-range (range# — addr data-capability address-capability size)` command returns the VMEbus master capabilities associated with the range number identified by `range#`. The value of `range#` may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation. The VMEbus address range being accessed is represented by the `addr-size` pair, where `addr` specifies the physical VMEbus address and `size` identifies the address range covered. The value of `data-capability` and `address-capability` may be one of the values listed in the table below.

The `vme-master-range! (addr data-capability address-capability size range# — )` command sets the VMEbus master capabilities associated with the range number identified by `range#`. The value of `range#` may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation. The VMEbus address range being accessed is represented by the `addr-size` pair, where `addr` specifies the physical VMEbus address and `size` identifies the address range covered. The value of `data-capability` and `address-capability` may be one of the values listed in the table below.

<table>
<thead>
<tr>
<th>value</th>
<th>data-capability</th>
<th>address-capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0002</td>
<td>cap-d8</td>
<td>cap-a16</td>
</tr>
<tr>
<td>0012</td>
<td>cap-d16</td>
<td>cap-a24</td>
</tr>
<tr>
<td>value</td>
<td>data-capability</td>
<td>address-capability</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>010₂</td>
<td>cap-d32</td>
<td>cap-a32</td>
</tr>
<tr>
<td>011₂</td>
<td>cap-blit</td>
<td>reserved</td>
</tr>
<tr>
<td>100₂</td>
<td>cap-mblt</td>
<td>cap-a64</td>
</tr>
<tr>
<td>101₂</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>110₂</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>111₂</td>
<td>reserved</td>
<td>reserved</td>
</tr>
</tbody>
</table>

`.vme-master-range (range# — )` displays the current settings of the VMEbus master interface associated with the range number `range#`. The value of `range#` may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`.vme-master-ranges (— )` displays the current settings of all register sets of the VMEbus master interface.

`.vme-cap (—)` displays the contents of the VMEbus Capability register.

`vme-master-map (range# — vaddr)` makes the physical address range, as defined by the contents of the range register set specified by the range number `range#`, available to the processor's virtual address space and returns the virtual address `vaddr`. Because the command obtains all information from the specific range register set, the particular range register must be initialized before.

`vme-master-unmap (vaddr range# — )` removes the physical address range, as defined by the contents of the range register set specified by the range number `range#`, from the processor's virtual address space.

`addr, size>sbus-compare-code (address size — compare-code)` returns the SBus compare-code which corresponds with the given `address` and `size` pair.

`sbus-compare-code>addr, size (compare-code — address size)` converts the SBus compare-code to the corresponding `address` and `size` pair.

The examples on the following pages describe how to initialize the VMEbus interface for subsequent VMEbus master accesses.

The example below shows how to access a 1 MByte area within the extended address space (A32) of the VMEbus beginning at address 4080.0000₁₆. The register set associated with the range number zero (`range#` is 0) is used to access the VMEbus area mentioned above. The first commands initializes the VMEbus master interface. It sets the data- and address
capabilities, as well as the VMEbus address and the size of the area being accessed. The *data capability* is defined using the predefined constant `cap-d32` which enables the VMEbus master interface to access bytes (8bit data), half-words (16bit data), and words (32-bit data) within the VMEbus area. The *address capability* is defined using the predefined constant `cap-a32` that enables the VMEbus interface to access the *extended* address space (A32) of the VMEbus.

The SBus slave interface is initialized by the second command which specifies that the VMEbus is accessed when the SBus slot one (1) is being accessed at offsets A0.0000 to BF.FFFF which corresponds to the VMEbus addresses in the range 4080.0000 to 409F.FFFF of the *extended* address space (A32).

```
ok h# 4080.0000 cap-d32 cap-a32 1Meg 2 * 0 vme-master-range!
ok 1Meg d# 10 * 1 1Meg 2 * 0 sbus-slave-range!
ok 0 vme-master-ena
ok 0 vme-master-map value vmebus
ok
```

Finally, the third command enables any access to the VMEbus. The fourth command maps the physical address area to be accessed in order to address the VMEbus to the virtual address space of the processor and stores the virtual address in the variable `vmebus`. This variable may be used to access the VMEbus area using the commands to read and write data provided by OpenBoot.

```
ok vmebus 0 vme-master-unmap
ok
```

When the translation (SBus to VMEbus) defined by the contents of the register set associated with the range number zero is no longer used, then the memory mapped to the processor’s virtual address space to access the VMEbus must be released before the contents of this register set are modified. This has to be done with the command `vme-master-unmap` as stated above.

In the next example the VMEbus interface is initialized to allow accesses to the *standard* address space (A24) of the VMEbus beginnig at address 98.0000. The size of this area is 512KByte and the register set associated with the range number one (*range#* is 1) is used to access this VMEbus area.

The first commands initializes the VMEbus master interface. It sets the data- and address capabilities, as well as the VMEbus address and the size of the area being accessed. The *data capability* is defined using the predefined constant `cap-d16` which enables the VMEbus master interface to access bytes (8bit data), and half-words (16bit data) within the VMEbus area. The *address capability* is defined using the predefined constant `cap-a24` that enables the VMEbus interface to access the *standard* address space (A24) of the VMEbus.

The SBus slave interface is initialized by the second command which specifies that the VMEbus is accessed when the SBus slot one (1) is being accessed at offsets 120.0000 to 127.FFFF which corresponds to the VMEbus addresses in the range 98.0000 to 9F.FFFF of the *standard* address space (A24).
Finally, the third command enables any access to the VMEbus. The fourth command maps the physical address area to be accessed in order to address the VMEbus to the virtual address space of the processor and stores the virtual address in the variable `vmebus`. This variable may be used to access the VMEbus area using the commands to read and write data provided by OpenBoot.

```
ok 1 vme-master-map value vmebus
```

When the translation (SBus to VMEbus) defined by the contents of the register set associated with the range number zero is no longer used, then the memory mapped to the processor’s virtual address space to access the VMEbus must be released before the contents of this register set are modified. This has to be done with the command `vme-master-unmap` as stated above.

The last example describes how to initialized the VMEbus interface to allow accesses to the short address space (A16) of the VMEbus begining at address 0000<sub>16</sub>. The size of this area is 64KByte and therefore covers the entire short address space. The register set associated with the range number two (`range#` is 2) is used to access this VMEbus area.

Again, the first commands initializes the VMEbus master interface. It sets the data- and address capabilities, as well as the VMEbus address and the size of the area being accessed. The data capability is defined using the predefined constant `cap-d8` which limits the VMEbus master interface to access only bytes (8bit data) within the VMEbus area. The address capability is defined using the predefined constant `cap-a16` that enables the VMEbus interface to access the standard address space (A16) of the VMEbus.

The SBus slave interface is initialized by the second command which specifies that the VMEbus is accessed when the SBus slot one (1) is being accessed at offsets 400.0000<sub>16</sub> to 400.FFFF<sub>16</sub> which corresponds to the VMEbus addresses in the range 0000<sub>16</sub> to FFFF<sub>16</sub> of the short address space (A16).

```
ok h# 000000 cap-d8 cap-a16 h# 1.0000 2 vme-master-range!
ok 1Meg d# 64 * 3 h# 1.0000 2 sbus-slave-range!
ok 2 vme-master-ena
ok 2 vme-master-map value vmebus
```

Finally, the third command enables any access to the VMEbus. The fourth command maps the physical address area to be accessed in order to address the VMEbus to the virtual address space of the processor and stores the virtual address in the variable `vmebus`. This variable may be used to access the VMEbus area using the commands to read and write data provided by OpenBoot.
When the translation (SBus to VMEbus) defined by the contents of the register set associated with the range number zero is no longer used, then the memory mapped to the processor’s virtual address space to access the VMEbus must be released before the contents of this register set are modified. This has to be done with the command `vme-master-unmap` as stated above.

Assumed the first three register sets have been used to access the VMEbus address spaces as described in the examples above, then the following command may be used to display the settings of the registers sets:

```
ok 2 vme-master-map value vmebus
ok .vme-master-ranges
```
The following commands are available to control the various operating modes of the SPARC FGA-5000 SBus interface.

`sbus-burst-length@ (— #burst-length) returns the maximum length of an SBus burst that is generated by the SPARC FGA-5000. The value of #burst-length is in the range zero through three. Each value specifies one of four possible burst lengths as stated in the table below.

`sbus-burst-length! (#burst-length — ) sets the maximum length of an SBus burst that is generated by the SPARC FGA-5000. The value of #burst-length may be in the range zero through three. Each value specifies one of four possible burst lengths as stated in the table below. The command considers only the least significant two bits of the value #burst-length.

<table>
<thead>
<tr>
<th>#burst-length</th>
<th>Burst Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8-byte burst</td>
</tr>
<tr>
<td>1</td>
<td>16-byte burst</td>
</tr>
<tr>
<td>2</td>
<td>32-byte burst</td>
</tr>
<tr>
<td>3</td>
<td>64-byte burst</td>
</tr>
</tbody>
</table>

`sbus-master-read-stop-point@ (— #read-stop-point) returns the SBus master read stop point currently in effect. The value of #read-stop-point is in the range zero through three. Each value specifies one of four possible read stop points as stated in the table below.

`sbus-master-read-stop-point! (#read-stop-point — ) sets the SBus master read stop point used by the SPARC FGA-5000. The value of #read-stop-point may be in the range zero through three. Each value specifies one of four possible read stop points as stated in the table below. The command considers only the least significant two bits of the value #read-stop-point.

<table>
<thead>
<tr>
<th>#read-stop-point</th>
<th>Read Stop Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Stop at 8-byte boundary</td>
</tr>
<tr>
<td>1</td>
<td>Stop at 16-byte boundary</td>
</tr>
<tr>
<td>2</td>
<td>Stop at 32-byte boundary</td>
</tr>
<tr>
<td>3</td>
<td>Stop at 64-byte boundary</td>
</tr>
</tbody>
</table>
sbus-retry-time@ (- retry-time) returns the number of SBus clocks before an SBus cycle is terminated with a retry by the SPARC FGA-5000. The value of retry-time is in the range zero through 255 and specifies the number of SBus clocks.

sbus-retry-time! ( retry-time - ) sets the number of SBus clocks before an SBus cycle is terminated with a retry by the SPARC FGA-5000. The value of retry-time may be in the range zero through 255 and specifies the number of SBus clocks. The command treats the value of retry-time as a modulo 256 number. When the command is called it verifies whether the given number of SBus clocks falls below the limit specified by min-retry-time. If this value falls below the given limit, then the commands uses the value of min-retry-time instead. This ensures that the SBus interface is operating properly.

sbus-rerun! ( true | false ) enables or disables the SPARC FGA-5000's capability to generate reruns on the SBus. When the value true is passed to the command the SPARC FGA-5000 will initiate SBus rerun if necessary. Otherwise — the value false is passed to the command — the SPARC FGA-5000's capability to initiate SBus reruns is disabled.

sbus-rerun-ena ( - ) enables the SPARC FGA-5000's capability to generate reruns on the SBus.

sbus-rerun-dis ( - ) disables the SPARC FGA-5000's capability to generate reruns on the SBus.

sbus-rerun-limit@ ( - #rerun-limit ) returns the number of reruns before the SPARC FGA-5000 terminates an SBus cycle with an error. The value of #rerun-limit is in the range zero through 255 and specifies the number of reruns.

sbus-rerun-limit! ( #rerun-limit - ) sets the number of reruns before the SPARC FGA-5000 terminates an SBus cycle with an error. The value of #rerun-limit may be in the range zero through 255 and specifies the number of reruns. The command treats the value of #rerun-limit as a modulo 256 number. When the command is called it verifies whether the given number of reruns falls below the limit specified by min-rerun-limit. If this value falls below the given limit, then the commands use the value of min-rerun-limit instead. This ensures that the SBus interface is operating properly.

sbus-burst-ena ( - ) enables the SPARC FGA-5000's capability to transfer data using SBus burst transfers.

sbus-burst-dis ( - ) disables the SPARC FGA-5000's capability to transfer data using SBus burst transfers.
sbus-hidden-arb-ena (—) enables the SPARC FGA-5000’s capability to perform hidden arbitration.

sbus-hidden-arb-dis (—) disables the SPARC FGA-5000’s capability to perform hidden arbitration.

sbus-split-flow-ena (—) enables split flowthrough.

sbus-split-flow-dis (—) disables split flowthrough.

sbus-split-ena (—) enables the SPARC FGA-5000’s capability to split SBus cycles.

sbus-split-dis (—) disables the SPARC FGA-5000’s capability to split SBus cycles.

.sbus-cap (—) displays the current contents of the SBus Master Capability Register as shown below:

ok .sbus-cap
Split: 1 Split Flow: 1 Arbiter: 1 Burst: 1
Master Read Stop Point: 32 bytes Max. Burst Length: 32 bytes
ok

.sbus-retry-time-ctrl (—) displays the current contents of the SPARC FGA-5000’s SBus Retry Time Control Register as stated below:

ok .sbus-retry-time-ctrl
Retry time: 10
ok

.sbus-rerun-limit-ctrl (—) displays the current contents of the SBus Rerun Limit Control Register as depicted below:

ok .sbus-rerun-limit-ctrl
Enable Reruns: 0 Rerun limit: 255
ok
4.2.9 VMEbus Slave Interface

The SPARC FGA-5000 provides three sets of registers to control any VMEbus slave access. Each set may be used to make a certain slave address range — standard- (A24) or extended (A32) slave address range — available to the VMEbus’ address space. A register set is identified by a unique number, the *range number* (*range#*), in the range zero through two. Only the A24 and A32 slave mode allows a VMEbus master to access the memory of the SPARC CPU-5V. In the A16 slave mode all VMEbus master accesses are limited to the registers of the SPARC FGA-5000 which are accessible from the VMEbus.

When the VMEbus interface is being accessed from the VMEbus, then the part of the SPARC FGA-5000 connected with the VMEbus is considered as VMEbus slave device. Whereas the part of the SPARC FGA-5000 that is connected with the SBus is operating as the SBus master. This fact is reflected in the names of the commands available to control the VMEbus master interface.

The commands listed and described in the following are available to initialize and control the A16 VMEbus slave interface:

vme-a16-slave-ena (-) enables the capability to access the SPARC FGA-5000 registers from the VMEbus in the *short* address space (A16).

vme-a16-slave-dis (-) disables the capability to access the SPARC FGA-5000 registers from the VMEbus in the *short* address space (A16).

vme-a16-slave-addr@ (— *addr*) returns the 16-bit address *addr* at which the registers of the SPARC FGA-5000 are accessible within the *short* address space (A16).

vme-a16-slave-addr! (addr — ) defines the 16-bit address *addr* at which the registers of the SPARC FGA-5000 are accessible within the *short* address space (A16). The least significant nine bits of the address *addr* are ignored by the command — the command treats them as if they are cleared —, because the SPARC FGA-5000 is only accessible from the VMEbus beginning at 512 Byte boundaries.

The commands listed and described in the following are available to initialize and control the A24 and A32 VMEbus slave interface:

vme-slave-ena ( *range#* — ) enables the address decoding associated with the range number *range#* to allow accesses from the VMEbus. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

vme-slave-dis ( *range#* — ) disables the address decoding associated with the range number *range#* to allow accesses from the VMEbus. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.
vrne-slave-wp-ena (range# — ) enables write posting within the VMEbus slave address range associated with the range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

vrne-slave-wp-dis (range# — ) disables write posting within the VMEbus slave address range associated with the range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

/vme-a24-range (range# — size ) returns the size of the standard (A24) slave interface associated with the range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

sbus-a24-master-range@ (range# — vaddr size) returns the SBus master parameters associated with the A24 slave interface identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access. The parameters vaddr and size identify the virtual address range within the SBus, into which all A24 slave accesses are translated.

sbus-a24-master-range! (vaddr size range# — ) defines the SBus master parameters associated with the A24 slave interface identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access. The parameters vaddr and size identify the virtual address range within the SBus, into which all A24 slave accesses are translated.

vme-a24-slave-range@ (range# — paddr size) returns the VMEbus base address paddr and the size size of the A24 slave window associated with the range identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

vme-a24-slave-range! (paddr size range# — ) sets the VMEbus base address paddr and the size size of the A24 slave window associated with the range identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.
/vme-a32-range (range# — size) returns the size of the extended (A32) slave interface associated with the range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

sbus-a32-master-range@ (range# — vaddr size) returns the SBus master parameters associated with the A32 slave interface identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access. The parameters vaddr and size identify the virtual address range within the SBus, into which all A32 slave accesses are translated.

sbus-a32-master-range! (vaddr size range# — ) defines the SBus master parameters associated with the A32 slave interface identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access. The parameters vaddr and size identify the virtual address range within the SBus, into which all A32 slave accesses are translated.

vme-a32-slave-range@ (range# — paddr size) returns the VMEbus base address paddr and the size size of the A32 slave window associated with the range identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

vme-a32-slave-range! (paddr size range# — ) sets the VMEbus base address paddr and the size size of the A32 slave window associated with the range identified by range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

.vme-slave-range (range# — ) displays the current settings of the VMEbus slave interface associated with the range number range#. The value of range# may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

.vme-slave-ranges () displays the current settings of all register sets controlling any VMEbus slave access.

addr, size>vme-compare-code (address size — compare-code) returns the VMEbus compare-code which corresponds with the given address and size pair.

vme-compare-code>addr, size (compare-code — address size) converts the VMEbus compare-code to the corresponding address and size pair.
The following example lists all steps to be taken, to initialize the VMEbus interface for A32 accesses from the VMEbus beginning at address 2340.0000_{16} and ranging to 235F.FFFF_{16}. The register set associated with the *range number* zero (0) is used to control this particular VMEbus slave interface.

```
ok h# 2340.0000 1Meg 2 * 0 vme-a32-slave-range!
ok h# ffe0.0000 1Meg 2 * 0 sbus-a32-master-range!
ok h# 10.0000 obmem h# ffe0.0000 1Meg 2 * iomap-pages
ok 0 vme-slave-ena
```

As shown above the first command defines the VMEbus slave interface's base address and size of the slave window. The second command defines that any A32 access is translated to an access of the SBus beginning at SBus address FFE0.0000_{16}. And the third command creates all necessary entries within the IOMMU to translate the SBus access to an access of the on-board memory beginning at physical address 10.0000_{16}. Finally, the VMEbus slave interface is enabled using the fourth command.

The next example lists all steps to be taken, to initialize the VMEbus interface for A24 accesses from the VMEbus beginning at address C0.0000_{16} and ranging to CF.FFFF_{16}. The register set associated with the *range number* one (1) is used to control this particular VMEbus slave interface.

```
ok h# c0.0000 1Meg 1 vme-a24-slave-range!
ok h# fff0.0000 1Meg 1 sbus-a24-master-range!
ok h# 20.0000 obmem h# fff0.0000 1Meg iomap-pages
ok 1 vme-slave-ena
```

As shown above the first command defines the VMEbus slave interface's base address and size of the slave window. The second command defines that any A24 access is translated to an access of the SBus beginning at SBus address FFF0.0000_{16}. And the third command creates all necessary entries within the IOMMU to translate the SBus access to an access of the on-board memory beginning at physical address 20.0000_{16}. Finally, the VMEbus slave interface is enabled using the fourth command.
4.2.10 VMEbus Device Node

The OpenBoot device tree contains the device node for the VMEbus interface and is called "VME". It is a child device of the device node "/iommu" (The full pathname of the VMEbus interface device node is displayed by the command show-devs). The device alias vme is available as a shorthand representation of the VMEbus interface device-path.

The vocabulary of the VMEbus device includes the standard commands recommended for a hierarchical device. The words of this vocabulary are only available when the VMEbus device has been selected as shown below:

```
ok cd vme
ok words
selftest reset close open ...
... list of further methods of the device node
ok selftest .
0
ok device-end
ok
```

The example listed above, selects the VMEbus device and makes it the current node. The word words displays the names of the methods of the VMEbus device. And the third command calls the method selftest and the value returned by this method is displayed. The last command unselects the current device node, leaving no node selected.

The following methods are defined in the vocabulary of the VMEbus device:

- **open** (-true) prepares the package for subsequent use. The value **true** is always returned.
- **close** (-) frees all resources allocated by open.
- **reset** (-) puts the VMEbus Interface into quiet state.
- **selftest** (-error-number) performs a test of the VMEbus interface, and returns an error-number to report the course of the test. In the case that the device has been tested successfully the value zero is returned; otherwise it returns a specific error number to indicate a certain fail state.
- **decode-unit** (addr len — low high) converts the addr and len, a text string representation, to low and high which is a numerical representation of a physical address within the address space defined by the package.
- **map-in** (low high size — vaddr) creates a mapping associating the range of physical address beginning at low, extending for size bytes, within the package’s physical address space, with a processor virtual address vaddr.
map-out (vaddr size — ) destroys the mapping set by map-in at the given virtual address vaddr of length size.

dma-alloc (size — vaddr) allocates a virtual address range of length size bytes that is suitable for direct memory access by a bus master device. The memory is allocated according to the most stringent alignment requirements for the bus. The address of the acquired virtual memory vaddr is returned via the stack.

dma-free (vaddr size — ) releases a given virtual memory, identified by its address vaddr and size, previously acquired by dma-alloc.

dma-map-in (vaddr size cachable? — devaddr) converts a given virtual address range, specified by vaddr and size, into an address devaddr suitable for direct memory access on the bus. The virtual memory must be allocated already by dma-alloc. The SPARC CPU-5V does not support caching. Thus the cachable? flag is ignored.

dma-map-out (vaddr devaddr size — ) removes the direct memory access mapping previously created by dma-map-in.

dma-sync (vaddr devaddr size — ) synchronizes memory caches associated with a given direct memory access mapping, specified by its virtual address vaddr, the devaddr and its size that has been established by dma-map-in.
4.2.11 VMEbus NVRAM Configuration Parameters

The NVRAM configuration parameters listed below are available to control the initialisation and operation of the VMEbus Interface. The current state of these configuration parameters are displayed using the `printenv` command, and are modified using either the `setenv`, or the `set-default` command provided by OpenBoot.

`vme-sysfail-clear?` when the value of the configuration parameter is `true` the SYSFAIL* signal will be cleared by OpenBoot. In the case that the configuration parameter is `false` OpenBoot will not clear the SYSFAIL* signal, but the operating system which is loaded has to clear it. (default: `true`)

The state of this NVRAM configuration parameter is considered independent of the state of the `vme-init?` configuration parameter.

`vme-bus-timer?` controls whether the VMEbus transaction timer in the SPARC FGA-5000 is used to watch each VMEbus access. When the flag is `true` the transaction timer is enabled. If the flag is `false` the transaction timer is disabled. (default: `true`)

The state of this NVRAM configuration parameter is considered independent of the state of the `vme-init?` configuration parameter.

`vme-bus-timeout` contains the timeout value of the SPARC FGA-5000 VMEbus transaction timer and is a value in the range one to three. Each value selects a particular timeout period. Independent of the state of the configuration parameter `vme-bus-timer?` the timeout value is stored in the appropriate register. When the value of this configuration parameter is not in the range one through three, then the value three is used instead. (default: `3`)

The state of this NVRAM configuration parameter is considered independent of the state of the `vme-init?` configuration parameter.

`vme-slot#` specifies the logical VMEbus slot number assigned to the SPARC CPU-5V board. The value may be in the range one through 255, but preferably should be set in such a way that it corresponds with the number of an available VMEbus slot.

The state of this configuration parameter does not control whether the VMEbus interface is operating as system controller when the configuration parameter’s value is one. (default: `1`)

`vme-fair-req?` specifies whether the VMEbus requester operates in the `fair` mode when requesting the VMEbus. When the value of the configuration parameter is `true`, the VMEbus requester operates in the `fair` mode. Otherwise — the value of the configuration parameter is `false` — the requester does not operate not in the fair mode. (default: `true`
vme-init? controls whether the VMEbus interface is initialized by OpenBoot. When this flag is true the VMEbus interface is initialized according to the state of the NVRAM parameter listed below. In the case that the flag is false the VMEbus interface is not initialized. The VMEbus interface is initialized after OpenBoot set up the main memory. (default: true)

The state of the NVRAM configuration parameters listed in the following are only considered by OpenBoot when the configuration parameter vme-init? is true!

vme-intr1 controls whether the VMEbus interrupt request level 1 has to be enabled. When the value is 255 then the VMEbus interrupt request level 1 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 1 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 1 occurs. (default: 255_{10})

vme-intr2 controls whether the VMEbus interrupt request level 2 has to be enabled. When the value is 255 then the VMEbus interrupt request level 2 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 2 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 2 occurs. (default: 255_{10})

vme-intr3 controls whether the VMEbus interrupt request level 3 has to be enabled. When the value is 255 then the VMEbus interrupt request level 3 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 3 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 3 occurs. (default: 255_{10})

vme-intr4 controls whether the VMEbus interrupt request level 4 has to be enabled. When the value is 255 then the VMEbus interrupt request level 4 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 4 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 4 occurs. (default: 255_{10})

vme-intr5 controls whether the VMEbus interrupt request level 5 has to be enabled. When the value is 255 then the VMEbus interrupt request level 5 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 5 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 5 occurs. (default: 255_{10})
vme-intr6 controls whether the VMEbus interrupt request level 6 has to be enabled. When the value is 255 then the VMEbus interrupt request level 6 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 6 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 6 occurs. (default: 255₁₀)

vme-intr7 controls whether the VMEbus interrupt request level 7 has to be enabled. When the value is 255 then the VMEbus interrupt request level 7 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 7 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 7 occurs. (default: 255₁₀)

vme-sysfail-assert? controls whether a nonmaskable interrupt is generated upon the assertion of the VMEbus signal SYSFAIL*. When the flag is true an interrupt handler, dealing with this interrupt, is installed and the ability to generate a nonmaskable interrupt upon the assertion of the SYSFAIL* signal is enabled. In the case that the flag is false the ability to generate a nonmaskable interrupt upon the assertion of the SYSFAIL* signal is enabled. (default: false)

vme-sysfail-negate? controls whether a nonmaskable interrupt is generated upon the negation of the VMEbus signal SYSFAIL*. When the flag is true an interrupt handler, dealing with this interrupt, is installed and the ability to generate a nonmaskable interrupt upon the negation of the SYSFAIL* signal is enabled. In the case that the flag is false the ability to generate a nonmaskable interrupt upon the negation of the SYSFAIL* signal is enabled. (default: false)

vme-acfail-assert? controls whether a nonmaskable interrupt is generated upon the assertion of the VMEbus signal ACFAIL*. When the flag is true an interrupt handler, dealing with this interrupt, is installed and the ability to generate a nonmaskable interrupt upon the assertion of the ACFAIL* signal is enabled. In the case that the flag is false the ability to generate a nonmaskable interrupt upon the assertion of the ACFAIL* signal is enabled. (default: false)

vme-ibox-addr the least significant 16 bits of this 32-bit configuration parameter define the address at which the interrupt box (IBOX) of the SPARC FGA-5000 is accessible within the short address space (A16). Only the least significant 16 bits of this configuration parameter are considered, and the state of the remaining bits is ignored. Independent of the configuration parameter vme-ibox-ena? OpenBoot will set the address of the IBOX. (default: 0₁₆)

vme-ibox-ena? indicates whether the interrupt box (IBOX), accessible in the short (A16) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is true then the IBOX is enabled. In the case that the NVRAM configura-
tion parameter is false the IBOX is not enabled.
The default value of this NVRAM configuration parameter is false.

fmb-init? controls whether the FMB system is initialized by OpenBoot. When this flag is true the FMB system is initialized according to the state of the NVRAM parameter listed below. In the case that the flag is false the FMB system is not initialized. The FMB system is initialized only during the initialization of the VMEbus interface, which means that the vme-init? configuration parameter must be true, in order to set up the FMB system. (default: true)

fmb-slot# specifies the logical slot number assigned to the FMB channels of the SPARC CPU-5V board. The value may be in the range one through 21, and preferably should be set in such a way that it corresponds with the number of an available VMEbus slot. (default: 110)

fmb-addr specifies the address — the most significant eight bits of a 32-bit address — where the FMB system resides in the extended address space (A32) of the VMEbus. (default: fa16)

The NVRAM configuration parameters listed below are associated with the slave interface accessible in the short (A16) address range.

vme-a16-slave-addr specifies the base address of the slave interface accessible in the short (A16) address range of the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a16-slave-size specifies the size of the memory which is made available to the short (A16) address range of the VMEbus. When the value of this configuration parameter is zero OpenBoot will not initialize the slave interface, even if the vme-a16-slave-ena? configuration parameter is true!
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a16-slave-ena? indicates whether the slave interface, accessible in the short (A16) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is true then the VMEbus slave interface is enabled. In the case that the NVRAM configuration parameter is false the VMEbus slave interface is not enabled, and any attempt to access the VMEbus slave interface from the VMEbus will lead to an error termination on the VMEbus.
The default value of this NVRAM configuration parameter is false.

In the case that the NVRAM configuration parameter vme-init? is true and the OpenBoot will initialize the slave interface according to the configuration parameters described above. When the vme-a16-slave-ena? configuration parameter is true, then OpenBoot will
initialize the VMEbus slave interface according to the NVRAM configuration parameters vme-a16-slave-addr and vme-a16-slave-size. It will provide the required amount of physical on-board memory and builds up the necessary MMU and IOMMU settings to make the memory available to the VMEbus. The virtual base address of the physical on-board memory provided for VMEbus slave accesses is stored in the variable vme-a16-slave-mem.

Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the slave interface completely according to the NVRAM configuration parameters associated with the slave interface.

In addition, this mechanism allows to report the parameters of the slave interface to an operating system loaded, which in turn provides its own memory and the corresponding MMU and IOMMU settings. In this case the VMEbus device driver is responsible for the access to the slave interface from the VMEbus. In general, the configuration parameter vme-a16-slave-ena? must be set to false to prevent OpenBoot from initialising and enabling the slave interface when an operating system will be loaded. Supposed that the slave interface is initialized and enabled by OpenBoot prior to loading the operating system, any access from the VMEbus to the slave interface while loading the operating system may alter memory and cause severe damage.

Note! The SPARC CPU-5V does not provide the ability to access its on-board memory from the VMEbus within the short (A16) address range. Therefore, the NVRAM configuration parameters associated with the A16 slave interface, control the access to the registers of the SPARC FGA-5000, which are accessible within the short address range. The configuration parameter vme-a16-slave-size is not of any importance and will be ignored.

The NVRAM configuration parameters listed below are associated with the slave interface accessible in the standard (A24) address range.

vme-a24-slave-addr specifies the base address of the slave interface accessible in the standard (A24) address range of the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a24-slave-size specifies the size of the memory which is made available to the standard (A24) address range of the VMEbus. When the value of this configuration parameter is zero OpenBoot will not initialize the slave interface, even if the vme-a24-slave-ena? configuration parameter is true!
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a24-slave-ena? indicates whether the slave interface, accessible in the standard (A24) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is true then the VMEbus slave interface is enabled. In the case that the NVRAM configuration parameter is false the VMEbus slave interface is not enabled, and any attempt to access the VMEbus slave interface from the VMEbus will lead to an error termination on the VMEbus.
The default value of this NVRAM configuration parameter is false.
In the case that the NVRAM configuration parameter vme-init? is true and the OpenBoot will initialize the slave interface according to the configuration parameters described above. When the vme-a24-slave-ena? configuration parameter is true, then OpenBoot will initialize the VMEbus slave interface according to the NVRAM configuration parameters vme-a24-slave-addr and vme-a24-slave-size. It will provide the required amount of physical on-board memory and builds up the necessary MMU and IOMMU settings to make the memory available to the VMEbus. The virtual base address of the physical on-board memory provided for VMEbus slave accesses is stored in the variable vme-a24-slave-mem.

Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the slave interface completely according to the NVRAM configuration parameters associated with the slave interface.

In addition, this mechanism allows to report the parameters of the slave interface to an operating system loaded, which in turn provides its own memory and the corresponding IOMMU settings. In this case the VMEbus device driver is responsible for the access to the slave interface from the VMEbus. In general, the configuration parameter vme-a24-slave-ena? must be set to false to prevent OpenBoot from initialising and enabling the slave interface when an operating system will be loaded. Supposed that the slave interface is initialized and enabled by OpenBoot prior to loading the operating system, any access from the VMEbus to the slave interface while loading the operating system may alter memory and cause severe damage.

The NVRAM configuration parameters listed below are associated with the slave interface accessible in the extended (A32) address range.

vme-a32-slave-addr specifies the base address of the slave interface accessible in the extended (A32) address range of the VMEbus.

The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a32-slave-size specifies the size of the memory which is made available to the extended (A32) address range of the VMEbus. When the value of this configuration parameter is zero OpenBoot will not initialize the slave interface, even if the vme-a24-slave-ena? configuration parameter is true!

The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a32-slave-ena? indicates whether the slave interface, accessible in the extended (A32) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is true then the VMEbus slave interface is enabled. In the case that the NVRAM configuration parameter is false the VMEbus slave interface is not enabled, and any attempt to access the VMEbus slave interface from the VMEbus will lead to an error termination on the VMEbus.

The default value of this NVRAM configuration parameter is false.

In the case that the NVRAM configuration parameter vme-init? is true and the OpenBoot
will initialize the slave interface according to the configuration parameters described above. When the `vme-a32-slave-ena?` configuration parameter is `true`, then OpenBoot will initialize the VMEbus slave interface according to the NVRAM configuration parameters `vme-a16-slave-addr` and `vme-a32-slave-size`. It will provide the required amount of physical on-board memory and builds up the necessary MMU and IOMMU settings to make the memory available to the VMEbus. The `virtual` base address of the physical on-board memory provided for VMEbus slave accesses is stored in the variable `vme-a32-slave-mem`.

Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the slave interface completely according to the NVRAM configuration parameters associated with the slave interface.

In addition, this mechanism allows to report the parameters of the slave interface to an operating system loaded, which in turn provides its own memory and the corresponding IOMMU settings. In this case the VMEbus device driver is responsible for the access to the slave interface from the VMEbus. In general, the configuration parameter `vme-a32-slave-ena?` must be set to `false` to prevent OpenBoot from initialising and enabling the slave interface when an operating system will be loaded. Suppose that the slave interface is initialized and enabled by OpenBoot prior to loading the operating system, any access from the VMEbus to the slave interface while loading the operating system may alter memory and cause severe damage.

The NVRAM configuration parameters listed below are associated with the master interface to access the `short` (A16) address range.

`vme-a16-master-addr` specifies the base address of the `short` (A16) address range to be accessed on the VMEbus.

The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a16-master-size` specifies the size of the area in the `short` (A16) address range of the VMEbus which will be accessed. When the value of this configuration parameter is zero OpenBoot will not initialize the master interface, even if the `vme-a16-master-ena?` configuration parameter is `true`! If the specified size exceeds the size of the `short` (A16) address range, then it limits the specified size to 64 Kbyte. Due to the capabilities of the SPARC FGA-5000 OpenBoot will always adjust the specified size to 64 Kbyte.

The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a16-master-ena?` indicates whether the master interface, to access the `short` (A16) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the VMEbus master interface is enabled. In the case that the NVRAM configuration parameter is `false` the VMEbus master interface is not enabled.

The default value of this NVRAM configuration parameter is `false`. 

---


OpenBoot

OPEN SOURCE COMPUTER Page 139
In the case that the NVRAM configuration parameter `vme-init?` is true OpenBoot will initialize the master interface according to the configuration parameters described above. When the `vme-a16-master-ena?` configuration parameter is true, then OpenBoot will initialize the necessary registers in the master interface and provides the virtual memory to access the VMEbus. The *virtual* base address necessary to access the VMEbus is stored in the variable `vme-a16-master-mem`.

Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the master interface completely according to the NVRAM configuration parameters associated with the master interface.

In addition, this mechanism allows to report the parameters of the master interface to an operating system loaded, which in turn provides its own virtual memory to access the VMEbus. In this case the VMEbus device driver is responsible for providing the necessary virtual address range to access the VMEbus. In general, the configuration parameter `vme-a16-master-ena?` must be set to `false` to prevent OpenBoot from initialising and enabling the master interface when an operating system will be loaded.

The NVRAM configuration parameters listed below are associated with the master interface to access the standard (A24) address range.

`vme-a24-master-addr` specifies the base address of the standard (A24) address range to be accessed on the VMEbus. The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a24-master-size` specifies the size of the area in the standard (A24) address range of the VMEbus which will be accessed. When the value of this configuration parameter is zero OpenBoot will not initialize the master interface, even if the `vme-a24-master-ena?` configuration parameter is `true`. If the specified size exceeds the size of the standard (A24) address range, then it limits the specified size to 16 Mbyte. The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a24-master-ena?` indicates whether the master interface, to access the standard (A24) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the VMEbus master interface is enabled. In the case that the NVRAM configuration parameter is `false` the VMEbus master interface is not enabled. The default value of this NVRAM configuration parameter is `false`.

In the case that the NVRAM configuration parameter `vme-init?` is true OpenBoot will initialize the master interface according to the configuration parameters described above. When the `vme-a24-master-ena?` configuration parameter is true, then OpenBoot will initialize the necessary registers in the master interface and provides the virtual memory to access the VMEbus. The *virtual* base address necessary to access the VMEbus is stored in the variable `vme-a24-master-mem`.

Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the master interface completely according to the
NVRAM configuration parameters associated with the master interface.

In addition, this mechanism allows to report the parameters of the master interface to an operating system loaded, which in turn provides its own virtual memory to access the VMEbus. In this case the VMEbus device driver is responsible for providing the necessary virtual address range to access the VMEbus. In general, the configuration parameter `vme-a24-master-ena?` must be set to `false` to prevent OpenBoot from initializing and enabling the master interface when an operating system will be loaded.

The NVRAM configuration parameters listed below are associated with the master interface to access the *extended* (A32) address range.

- `vme-a32-master-addr` specifies the base address of the *extended* (A32) address range to be accessed on the VMEbus.
  - The default value of this 32-bit NVRAM configuration parameter is zero (0).

- `vme-a32-master-size` specifies the size of the area in the *standard* (A24) address range of the VMEbus which will be accessed. When the value of this configuration parameter is zero OpenBoot will not initialize the master interface, even if the `vme-a32-master-ena?` configuration parameter is `true`!
  - The default value of this 32-bit NVRAM configuration parameter is zero (0).

- `vme-a32-master-ena?` indicates whether the master interface, to access the *extended* (A32) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the VMEbus master interface is enabled. In the case that the NVRAM configuration parameter is `false` the VMEbus master interface is not enabled.
  - The default value of this NVRAM configuration parameter is `false`.

In the case that the NVRAM configuration parameter `vme-init?` is `true` OpenBoot will initialize the master interface according to the configuration parameters described above. When the `vme-a24-master-ena?` configuration parameter is `true`, then OpenBoot will initialize the necessary registers in the master interface and provides the virtual memory to access the VMEbus. The *virtual* base address necessary to access the VMEbus is stored in the variable `vme-a24-master-mem`.
Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the master interface completely according to the NVRAM configuration parameters associated with the master interface.

In addition, this mechanism allows to report the parameters of the master interface to an operating system loaded, which in turn provides its own virtual memory to access the VMEbus. In this case the VMEbus device driver is responsible for providing the necessary virtual address range to access the VMEbus. In general, the configuration parameter `vme-a32-master-ena?` must be set to `false` to prevent OpenBoot from initializing and enabling the master interface when an operating system will be loaded.
4.2.12 DMA Controller Support

The commands listed below are available to control the DMA controller of the SPARC FGA-5000, as well as to get information about the actual state of the DMA controller.

`dma-irq-map! (mapping —)` selects the interrupt to be generated by the DMA controller when the DMA process terminated successfully or due to an error. The parameter `mapping` defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of `mapping` may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table “Interrupt Mapping,” on page 109 lists all allowed mappings.

`dma-irq! (true | false)` enables or disables the interrupt to be generated by the DMA controller when the DMA process terminated successfully or due to an error. When the value `true` is passed to the command the interrupt is enabled. Otherwise — the value `false` is passed to the command — the interrupt is disabled.

`dma-ip? ( — true | false)` checks whether an interrupt is pending because a DMA process has been terminated. The value `true` is returned when an interrupt is pending due to the termination of a DMA process. Otherwise the value `false` is returned to indicate that no interrupt is pending.

`dma-ena ( —)` enables the DMA controller and starts a DMA process.

`dma-dis ( —)` disables the DMA controller and stops the DMA process currently running.

`dma-halt ( —)` halts the DMA process currently running.

`dma-resume ( —)` resumes the DMA process that has been halted before.

`dma-src-cap@ ( — data-capability address-capability)` returns the `data-capability` and `address-capability` currently defined for the source of the DMA process.

`dma-src-cap! ( data-capability address-capability —)` sets the `data-capability` and `address-capability` for the source of the DMA process.

The constants listed below are available to specify the `data-capability` and the `address-capability`:

<table>
<thead>
<tr>
<th>value</th>
<th>data-capability</th>
<th>address-capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>000₂</td>
<td>cap-d8</td>
<td>cap-a16</td>
</tr>
<tr>
<td>001₂</td>
<td>cap-d16</td>
<td>cap-a24</td>
</tr>
<tr>
<td>010₂</td>
<td>cap-d32</td>
<td>cap-a32</td>
</tr>
<tr>
<td>value</td>
<td>data-capability</td>
<td>address-capability</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>011_2</td>
<td>cap-blt</td>
<td>reserved</td>
</tr>
<tr>
<td>100_2</td>
<td>cap-mblt</td>
<td>reserved</td>
</tr>
<tr>
<td>101_2</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>110_2</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>111_2</td>
<td>reserved</td>
<td>reserved</td>
</tr>
</tbody>
</table>

dma-dest-cap@ (— data-capability address-capability ) returns the data-capability and address-capability currently defined for the destination of the DMA process.

dma-dest-cap! ( data-capability address-capability — ) sets the data-capability and address-capability for the destination of the DMA process.
The constants listed below are available to specify the data-capability and the address-capability:

<table>
<thead>
<tr>
<th>value</th>
<th>data-capability</th>
<th>address-capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>000_2</td>
<td>cap-d8</td>
<td>cap-a16</td>
</tr>
<tr>
<td>001_2</td>
<td>cap-d16</td>
<td>cap-a24</td>
</tr>
<tr>
<td>010_2</td>
<td>cap-d32</td>
<td>cap-a32</td>
</tr>
<tr>
<td>011_2</td>
<td>cap-blt</td>
<td>reserved</td>
</tr>
<tr>
<td>100_2</td>
<td>cap-mblt</td>
<td>reserved</td>
</tr>
<tr>
<td>101_2</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>110_2</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>111_2</td>
<td>reserved</td>
<td>reserved</td>
</tr>
</tbody>
</table>

dma-count@ (— transfer-count ) returns the current state of the transfer count. The value transfer-count indicates the number of bytes to be transfer by the DMA controller. Because the DMA controller only transfers a multiple of 32-bit data (longword, which is a word in the SPARC terminology), the command returns the appropriate number of words to be transferred.

dma-count! ( transfer-count — ) sets the number of bytes — transfer-count — to be transferred by the DMA controller.
Because the DMA controller only transfers a multiple of 32-bit data (longword, which is a word in the SPARC terminology), the command calculates the appropriate number of words to be transferred. The transfer-count is considered to be a modulo 4 Mbyte less four bytes number.
dma-running? (—true | false ) checks whether the DMA controller is in the *running* state. The value *true* is returned when the DMA controller is currently running. Otherwise the value *false* is returned to indicate that the DMA controller is disabled.

dma-waiting? (—true | false ) checks whether the DMA controller is in the *waiting* state. The value *true* is returned when the DMA controller is currently waiting, which means that it has been halted. Otherwise the value *false* is returned to indicate that the DMA controller is not waiting.

dma-normal-terminated? (—true | false ) checks whether the DMA process has been terminated successfully. It returns the value *true* when the DMA process has been terminated successfully. Otherwise the value *false* is returned to indicate that the DMA process has been terminated due to a fail state, or because the DMA process is still in progress.

dma-error-terminated? (—true | false ) checks whether the DMA process has been terminated unsuccessfully. It returns the value *true* when the DMA process has been terminated due to a fail state. Otherwise the value *false* is returned to indicate that the DMA process has been terminated due to normal termination, or because the DMA process is still in progress.

.dma-stat (—) displays the current state of the DMA Status Register.

```
ok .dma-stat
ERR: 3 NT: 0 HALT: 0 RUN: 0
ok
```

The fields **NT**, **HALT**, and **RUN** reflect the current state of the DMA controller. When the **NT** field is set to one (1), then the DMA controller terminated successfully (*normal termination*). In the case that the **HALT** field is set to one (1), then the DMA controller is halted — in general, this field is set along with the **RUN** field. The DMA controller is running when the **RUN** field is set to one (1). When one of the fields described previously is cleared (0), the DMA controller is not in the particular state.

Typically, the **ERR** field indicates the course of the DMA controller operation and may indicate the fail states listed in the table below:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error occurred on source bus</td>
</tr>
<tr>
<td>1</td>
<td>Error occurred on destination bus</td>
</tr>
<tr>
<td>2</td>
<td>No error termination</td>
</tr>
</tbody>
</table>
The following two commands used to initiate a DMA transfer do not set the data- and address capabilities of the source area and destination area. The capabilities **must** be appropriately set with the `dma-src-cap!` and `dma-dest-cap!` commands before the DMA transfer is started.

```
dma-mem>vme ( src-addr dest-addr count — true | false ) initiates a DMA transfer from the SBus to the VMEbus and **awaits** the termination of the DMA process. The amount of bytes given by `count` is transferred from `src-addr` — an address area on the SBus (virtual address) — to `dest-addr` — an address area on the VMEbus (physical address). The command returns the value `true` when all data have been transferred successfully. Otherwise the value `false` is returned to indicate that an error occurred during the DMA process. Because the DMA controller only transfers a multiple of 32-bit data (**longword**, which is a **word** in SPARC terminology), the command calculates the appropriate number of **words** to be transferred. Furthermore, the `count` is considered to be a modulo 4 Mbyte less four bytes number.

```
dma-vmem>mam ( src-addr dest-addr count — true | false ) initiates a DMA transfer from the VMEbus to the SBus and **awaits** the termination of the DMA process. The amount of bytes given by `count` is transferred from `src-addr` — an address area on the VMEbus (physical address) — to `dest-addr` — an address area on the SBus (virtual address). The command returns the value `true` when all data have been transferred successfully. Otherwise the value `false` is returned to indicate that an error occurred during the DMA process. Because the DMA controller only transfers a multiple of 32-bit data (**longword**, which is a **word** in the SPARC terminology), the command calculates the appropriate number of **words** to be transferred. Furthermore, the `count` is considered to be a modulo 4 Mbyte less four bytes number.
4.2.13 Mailboxes and Semaphores

The commands described in this section control the mailboxes, the semaphores, and the interrupt box (IBOX).

vme-mbox-take (mailbox# — true | false) takes the mailbox semaphore specified by mailbox# and returns the value true when the mailbox semaphore has been taken successfully. The value false is returned when the mailbox semaphore has been taken already.

The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vme-mbox-give (mailbox# — ) gives — releases — the mailbox semaphore specified by mailbox#.

The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vme-mbox-irq-map! (mapping mailbox# — ) selects the interrupt to be generated when the mailbox semaphore specified by mailbox# is taken. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the mailbox semaphore is taken. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

vme-mbox-irq-ena (mailbox# — ) allows the VMEbus interface to generate an interrupt when the mailbox specified by mailbox# is taken. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vme-mbox-irq-dis (mailbox# — ) disables the interrupt to be generated when the mailbox specified by mailbox# is taken. The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vme-mbox-ip? (mailbox — true | false) checks whether an interrupt is pending because the mailbox semaphores specified by mailbox# have been taken. The value true is returned when an interrupt is pending because the mailbox semaphore has been taken. Otherwise the value false is returned to indicate that no interrupt is pending.

The value of mailbox# may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

vme-sem-take (semaphore# — true | false) takes a semaphore specified by mailbox# and returns the value true when the semaphore has been taken successfully. The value
false is returned when the semaphore has been taken already.
The value of semaphore# may be one of the values in the range zero through 47. Each
value specifies one of the 48 Semaphore Registers.

vme-sem-give (semaphore# — ) gives — releases — the semaphore specified by semaphore#.
The value of semaphore# may be one of the values in the range zero through 47. Each
value specifies one of the 48 Semaphore Registers.

The Interrupt Box is only accessible from the VMEbus within the short address space (A16).
Any byte access — reading or writing — may lead the SPARC FGA-5000 to generate an
interrupt. The address of the interrupt box within the short address space may be any byte
location in the range 0000_{16} through FFFF_{16}.
The commands listed below are available to control and initialize the Interrupt Box.

vme-ibox-irq-map! (mapping — ) selects the interrupt to be generated when the inter­
rupt box is being accessed.
The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when
the interrupt box is accessed. The value of mapping may be one of the values in the
range zero through seven. Each value specifies one of the eight SPARC FGA-5000
interrupt request lines. The table “Interrupt Mapping.” on page 109 lists all allowed
mappings.

vme-ibox-irq-ena (— ) allows the VMEbus interface to generate an interrupt when the
interrupt box is accessed.

vme-ibox-irq-dis (— ) disables the interrupt to be generated when the interrupt box is
accessed.

vme-ibox-ip? (— true | false ) checks whether an interrupt is pending because the inter­
rupt box has been accessed. The value true is returned when an interrupt is pending
because the interrupt box has been accessed. Otherwise the value false is returned to
indicate that no interrupt is pending.

vme-ibox-ena (— ) enables the interrupt box.

vme-ibox-dis (— ) disables the interrupt box.

vme-ibox-addr@ (— addr ) returns the physical address addr of the interrupt box.

vme-ibox-addr! (addr — ) sets the physical address addr of the interrupt box.

As shown in the example below the first command sets the address of the interrupt box. The
interrupt box is accessible at the address 4002_{16} within the VMEbus short address space. An
SBus IRQ 5 is generated by the SPARC FGA-5000 whenever the interrupt box is accessed
from the VMEbus. The fourth command enables the interrupt box.
ok h# 4002 vme-ibox-addr!
ok 5 vme-ibox-irq-map!
ok vme-ibox-irq-ena
ok vme-ibox-ena
ok

4.2.14 FORCE Message Broadcast

The commands listed below are available to control the FORCE Message Broadcast (FMB) system and to obtain status information about the state of the FMB system.

fmb-super-only (true | false) allows or prevents the FMB message registers from being accessed in the non-privileged mode. When the value true is passed to the command the FMB message register is accessible in the privileged mode, as well as in the non-privileged mode. Otherwise — the value false is passed to the command — the FMB message registers are accessible in the privileged mode only.

fmb-ena (channel#) enables the FMB channel specified by channel#. The value of channel# may be one of the values in the range zero to one. Each value specifies one of the two FMB channels.

fmb-dis (channel#) disables the FMB channel specified by channel#. The value of channel# may be one of the values in the range zero to one. Each value specifies one of the two FMB channels.

fmb! ([true | false ]channel#) enables or disables the FMB channel specified by channel#. When the value true is passed to the command the FMB channel is enabled. Otherwise — the value false is passed to the command — the FMB channel is disabled.

fmb-slot@ (slot#) returns the slot number slot# assigned to the FMB channels.

fmb-slot! (slot#) assigns the slot number slot# to the FMB channels. The value of slot# may be one of the values in the range zero to 21. Each value specifies a specific slot.

fmb-addr@ (fmb-space) returns the most significant eight bits — the fmb-space — of the 32-bit VMEbus address the FMB will respond to when an FMB transaction on the VMEbus is detected.

fmb-addr! (fmb-space) sets the most significant eight bits — the fmb-space — of the 32-bit VMEbus address the FMB will respond to when an FMB transaction on the VMEbus is detected.
fmb-irq-map! (mapping channel# — ) selects the interrupt to be generated when the FMB message has been accepted, or rejected by the channel specified by channel#. The value of channel# may be one of the values in the range zero through one. Each value specifies one of the two FMB channels. The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the FMB message is accepted or rejected. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

fmb-irq! ([true | false] channel# — ) enables or disables the interrupt to be generated when either an FMB message has been accepted or rejected by the channel specified by channel#. The value of channel# may be one of the values in the range zero through one. Each value specifies one of the two FMB channels. When the value true is passed to the command the interrupt is enabled. Otherwise — the value false is passed to the command — the interrupt is disabled.

fmb-ip? (channel# — true | false) checks whether an interrupt is pending because an FMB message has been accepted or rejected by the channel specified by channel#. The value of channel# may be one of the values in the range zero through one. Each value specifies one of the two FMB channels. The value true is returned when an interrupt is pending. Otherwise the value false is returned to indicate that no interrupt is pending.

fmb-accepted-ip? (channel# — true | false) checks whether an interrupt is pending because an FMB message has been accepted by the channel specified by channel#. The value of channel# may be one of the values in the range zero through one. Each value specifies one of the two FMB channels. The value true is returned when an interrupt is pending because a message has been accepted. Otherwise the value false is returned to indicate that no interrupt is pending.

fmb-rejected-ip? (channel# — true | false) checks whether an interrupt is pending because an FMB message has been accepted by the channel specified by channel#. The value of channel# may be one of the values in the range zero through one. Each value specifies one of the two FMB channels. The value true is returned when an interrupt is pending because a message has been rejected. Otherwise the value false is returned to indicate that no interrupt is pending.

fmb-rejected-ip-clear (channel# — ) clears a pending message rejected interrupt generated by the channel specified by channel#.

fmb-msg@ (channel# — message true | false) fetches a message — a 32-bit data — from the FMB channel specified by channel#. The message and the value true are returned when an FMB is available. Otherwise the value false is returned to indicated that no FMB message is available.
fmb-msg! (message slot-list channel# — true | false ) sends the message — a 32-bit data — to all FMB channels identified by the slot-list and channel#. The value true is returned when the message has been sent out successfully. Otherwise the value false is returned to indicate that one or more FMB channels have rejected the message.

The value of channel# may be one of the values in the range zero through one. Each value specifies one of the two FMB channels.

The value of slot-list identifies the hosts participating in the FMB transaction. Each bit of the slot list is associated with a host identified by a unique FMB slot number. The first bit — bit 0 — relates to the host with the FMB slot number one (1); the second bit — bit 1 — relates to the host with the FMB slot number two (2); and so forth.

Because the FMB system allows only up to 21 hosts, the command considers only the least significant bits of the parameter slot-list (bit 0 through 20).

fmb-init (slot# fmb-space — ) performs all rudimentary steps to initialize the SPARC FGA-5000 in such a way that the subsequent FMB cycles are carried out using the fmb-msg! command.

The slot number slot# specifies the slot number the FMB channels are associated with.

The value of slot# may be one of the values in the range zero to 21. Each value specifies a specific slot.

The last available register set in the SPARC FGA-5000 is initialized to carry out an FMB cycle on the VMEbus within the appropriate VMEbus address area that has been specified by fmb-space. The parameter fmb-space defines the most significant eight bits (one of 256 16-Mbyte pages) of the VMEbus address where the FMB area is located. The capabilities of this VMEbus master range are A32/D32 and write posting is disabled. The variable fmb-va contains the virtual address to be accessed to execute an FMB cycle on the VMEbus.

The example below assigns the slot number 15_10 to the FMB channels available (all other hosts must have a different FMB slot number). The FMB address space is set to FA_16 which means that the FMB system is accessed when the address FAXX.XXXX_16 appears on the VMEbus address lines (the least significant 24 bits are used to select a specific FMB channel and specific hosts). And the second command enables the second FMB channel.

```
ok d# 15 h# fa fmb-init
ok true 1 fmb!
ok h# 1234AA55 h# 0010.800f 1 fmb-msg!
ok 1 fmb-msg@
ok .s 2drop
1234AA55 ffffffff
ok 1 fmb-msg@
ok .s drop
0
ok
```
Finally the message $1234.AA55_{16}$ is sent to the second FMB channel available on the hosts with the FMB slot number one, two, three, four, 15, and 20. Because the message is sent to the host with the FMB slot number 15 — the host that sent the message —, the message is read from the second FMB channel on the host, as shown by the fourth command. When the FMB channel is read again, and supposed the host did not receive another FMB message, the command `fmb-msg@` will return the value `false` to indicate that no more messages are available.

### 4.2.15 Diagnostic

The commands listed and described in this section are used to obtain various error status information from the SPARC FGA-5000 in the case of write posting errors and SBus errors.

- `wperr-irq-map! (mapping —)` selects the interrupt to be generated when a write posting error occurs on the SBus or VMEbus. The parameter `mapping` defines the interrupt asserted by the SPARC FGA-5000 when the write post error occurs. The value of `mapping` may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

- `wperr-irq! (true | false —)` enables or disables the interrupt to be generated when a write posting error occurs on the SBus or VMEbus. When the value `true` is passed to the command the interrupt is enabled. Otherwise — the value `false` is passed to the command — the interrupt is disabled.

- `vme-wperr-ip? ( — true | false )` checks whether an interrupt is pending because a write posting error occurred on the VMEbus. The value `true` is returned when an interrupt is pending due to a write posting error. Otherwise the value `false` is returned to indicate that no interrupt is pending.

- `sbus-wperr-ip? ( — true | false )` checks whether an interrupt is pending because a write posting error occurred on the SBus. The value `true` is returned when an interrupt is pending due to a write posting error. Otherwise the value `false` is returned to indicate that no interrupt is pending.

- `sbus-wperr-clear ( — error-addr )` reads the SBus Write Posting Error Address Register and returns the address `error-addr`.

- `vme-wperr-clear ( — error-addr )` reads the VMEbus Write Posting Error Address Register and returns the address `error-addr`.

- `slerr-irq-map! (mapping —)` selects the interrupt to be generated when a late error occurs on the SBus. The parameter `mapping` defines the interrupt asserted by the SPARC FGA-5000 when the late error occurs. The value of `mapping` may be one of
the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

\texttt{slerr-irq! (true \mid false \enspace \text{-})} enables or disables the interrupt to be generated when a late error occurs on the SBus. When the value \texttt{true} is passed to the command the interrupt is enabled. Otherwise — the value \texttt{false} is passed to the command — the interrupt is disabled.

\texttt{slerr-ip? (\enspace \text{-})} checks whether an interrupt is pending because a late error occurred on the SBus. The value \texttt{true} is returned when an interrupt is pending due to a late error. Otherwise the value \texttt{false} is returned to indicate that no interrupt is pending.

\texttt{slerr-clear (\enspace \text{-})} reads the SBus Late Error Address Register and returns the address \texttt{error-addr}.

\section*{4.2.16 Miscellanea}

The commands listed in this section are used to control miscellaneous functions in the SPARC FGA-5000.

\texttt{freeze-intr-mapping (\enspace \text{-})} prevents the SYSFAIL*, ACFAIL* and ABORT Interrupt Select and Enable Registers from being modified by setting the freeze bit in the Miscellaneous Control and Status Register. This mechanism is intended to prevent the appropriate Interrupt Control and Status Register from being modified after it has been initialized once.

\texttt{dtb-driver-ena (\enspace \text{-})} enables all VMEbus DTB drivers.

\texttt{dtb-driver-dis (\enspace \text{-})} disables all VMEbus DTB drivers.

\texttt{vme-timer-ena (\enspace \text{-})} enables the VMEbus transaction timer.

\texttt{vme-timer-dis (\enspace \text{-})} disables the VMEbus transaction timer.

\texttt{vme-timeout@ (\enspace \text{-})} returns the VMEbus transaction timer timeout value in use. The value of \texttt{timeout} may be one of the values in the range one through three. Each value identifies a particular timeout period as shown in the table below.

\texttt{vme-timeout! (timeout \enspace \text{-})} sets the VMEbus transaction timer timeout according to the given \texttt{timeout}. The value of \texttt{timeout} may be one of the values in the range one through three. When the value being specified is not in the range one through three, then the command selects the longest timeout period automatically. The values select a particular timeout period. The table below lists all possible values:
<table>
<thead>
<tr>
<th>timeout</th>
<th>$t_{\text{transaction-timeout}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32 us</td>
</tr>
<tr>
<td>2</td>
<td>128 us</td>
</tr>
<tr>
<td>3</td>
<td>512 us</td>
</tr>
</tbody>
</table>

Table 41: VMEbus Transaction Timer Timeout Values
4.3 Standard Initialization of the VMEbus Interface

Besides the initialization performed according to the state of the NVRAM configuration parameters, the VMEbus interface — mainly the SPARC FGA-5000 — is initialized as described in the subsections below.

4.3.1 SPARC FGA-5000 Registers

The registers of the SPARC FGA-5000 are accessible beginning at offset $0FFFF00_{16}$ within the SBus slot 5 and occupy the last 512 bytes in this slot ($0FFFF00_{16}$ ... $0FFFFFFF_{16}$). This corresponds with the physical address range $7FFFF00_{16}$ through $7FFFFFFF_{16}$.

The area in the range $0FE0000_{16}$ through $0FFFFDFF_{16}$ is available for any application. Preferably, this area may be used to access the standard (A24, max 16 MB) and short (A16, max 64 KB) address space of the VMEbus.

4.3.2 VMEbus Transaction Timer

The SPARC FGA-5000 contains a VMEbus transaction timer which is disabled after a RESET. This timer is enabled during the initialisation phase of OpenBoot and the transaction timeout period is set to the longest possible value (512 us).

4.3.3 SBus Rerun Limit

The SBus Rerun Limit counter, within the SPARC FGA-5000, is disabled to avoid any improper behaviour of the system.

4.3.4 Interrupts

The SPARC FGA-5000 is initialized in such a way that in the case of the occurrence of one of the events listed below, a nonmaskable interrupt (level 15 interrupt) is generated:

1.) Pressing the ABORT switch
4.3.5 SBus Slot 5 Address Map

SBus Slot 5

Offset

0FFF.FE00<sub>16</sub>

Available for VMEbus Accesses

SPARC FGA-5000 Registers

This area may be used to access the entire standard (A24) and short (A16) address space of the VMEbus.

0FE0.0000<sub>16</sub>

NCR 89C100
NCR 89C105

0000.0000<sub>16</sub>
4.4 System Configuration

4.4.1 Watchdog Timer

wd-ena ( — ) enables and starts the watchdog timer.

wd-dis ( — ) stops and disables the watchdog timer.

wd-timeout@ ( — timeout ) returns the watchdog timer’s reference value in use. The value of timeout may be one of the values in the range zero through seven. Each value identifies a particular timeout period as shown in the table below.

wd-timeout! ( timeout — ) sets the watchdog timer’s reference value for timeout according to the given timeout. The value of timeout may be one of the values in the range zero through seven. Only the least significant three bits of the value timeout are considered. The values select a particular timeout period. The table below lists all possible values:

<table>
<thead>
<tr>
<th>timeout</th>
<th>t_{wd-timeout-min}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>408 ms</td>
</tr>
<tr>
<td>1</td>
<td>1.68 s</td>
</tr>
<tr>
<td>2</td>
<td>6.7 s</td>
</tr>
<tr>
<td>3</td>
<td>26.8 s</td>
</tr>
<tr>
<td>4</td>
<td>1 min 48 s</td>
</tr>
<tr>
<td>5</td>
<td>7 min 9 s</td>
</tr>
<tr>
<td>6</td>
<td>28 min 38 s</td>
</tr>
<tr>
<td>7</td>
<td>1 h 54 min</td>
</tr>
</tbody>
</table>

Table 42: Watchdog Timer Timeout Values

wd-nmi-ena ( — ) allows an interrupt to generate when half of the watchdog time has expired.

wd-nmi-dis ( — ) disables the interrupt’s ability to generate when half of the watchdog time has expired.

wd-irq-map! ( mapping — ) selects the interrupt to be generated when half of the watch-
dog time has expired.
The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when half of the watchdog time has expired. The value of *mapping* may be one of the values in the range of zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping," on page 109 lists all allowed mappings.

`wd-nmi-clear` clears a pending interrupt caused by the watchdog timer when half of the watchdog time has expired.

`wd-ip?` checks whether an interrupt is pending due to an interrupt generated by the watchdog timer when half of the watchdog time has expired. The value `true` is returned when the interrupt is pending; otherwise the value `false` is returned.

`wd-restart` resets the watchdog timer and starts a new time count. In particular the command invokes one of the commands `vsi-wdt-restart@` or `vsi-wdt-restart!` to restart the watchdog timer.

The watchdog timer is started by the commands listed below:

```
ok 3 wd-timeout!
ok vsi-nmi wd-irq-map!
ok wd-nmi-ena
ok wd-ena
ok
```

In this example the *watchdog timer timeout* is set to 26.8 seconds, and a *nonmaskable* interrupt is generated whenever half of the watchdog time has expired. The OpenBoot already contains an interrupt handler dealing with the interrupt generated by the watchdog timer, and this interrupt handler increments an internal variable by one, whenever the watchdog timer emits an interrupt. The state of this variable is determined by:

```
ok wdnmi-occurred? ?
6
ok
```

This variable is cleared — set to zero — by

```
ok wdnmi-occurred? off
ok
```

`wd-reset?` determines whether a reset has been generated because the watchdog timer has expired. If a reset has been generated because the watchdog timer reached the timeout value, then the value `true` is returned; otherwise the value `false` is returned.
4.4.2 Watchdog Timer NVRAM Configuration Parameters

The NVRAM configuration parameters listed below are available to control the initialisation and operation of the watchdog timer. The current state of these configuration parameters are displayed using the printenv command, and are modified using either the setenv, or the set-default command provided by OpenBoot.

wd-ena? controls whether the watchdog timer has to be started. When the flag is true, then the watchdog timer is started after it has been initialized according to the configuration parameter wd-timeout. If the flag is false the watchdog timer is not started, but the watchdog timer registers are initialized according to the configuration parameter wd-timeout. (default: false)

wd-timeout contains the timeout value of the watchdog timer and is a value in the range 0 to 7. Each value selects a particular timeout period. Independent of the state of the configuration parameter wd-ena? the timeout value is stored in the appropriate watchdog timer register. (default: 7_{10})

4.4.3 Abort Switch

abort-switch? (— true | false ) determines the current state of the abort switch. The value true is returned when the abort switch is pressed. And the value false is returned when the abort switch is released.

abort-irq-map! (mapping — ) selects the interrupt to be generated when the abort switch is pressed.

The parameter mapping defines the interrupt asserted by the SPARC FGA-5000 when the abort switch is pressed. The value of mapping may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

abort-nmi-ena (—) allows an interrupt to generate when the abort switch is pressed.

abort-nmi-dis (—) disables the interrupt's ability to generate when the abort switch is being pressed.

abort-ip? (— true | false ) checks whether an interrupt is pending because the abort switch has been pressed. The value true is returned when the interrupt is pending; otherwise the value false is returned.

abort-nmi-clear (—) clears a pending interrupt caused by the abort switch.
4.4.4 Abort Switch NVRAM Configuration Parameter

The NVRAM configuration parameter listed below is available to control the initialisation and operation of the abort switch. The current state of these configuration parameters are displayed using the printenv command, and are modified using either the setenv, or the setdefault command provided by OpenBoot.

abort-ena? controls whether the abort switch has to be enabled. When this flag is true the abort switch is enabled and has the same effect as pressing the STOP-A key on an available keyboard. If the flag is false then the abort switch is disabled. (default: false)

4.4.5 LEDs, Seven-Segment Display and Rotary Switch

The commands described below are available to control the seven-segment LED display, the user LEDs, and are used to retrieve information about the state of the rotary switch.

diag-led! (byte — ) stores the data byte passed to the command in the register used to control the seven-segment display.

>7-seg-code (u — 7-seg-code ) converts the value u to its corresponding seven-segment code 7-seg-code. Only the least significant four bits of the value u are considered.

led! (colour freq led# — ) controls the user LED identified by led#. The value of led# may be either zero or one. The value zero specifies the first user LED, and the value one specifies the second user LED. The command only considers the state of the bit 0 of the value led#.

The parameters colour and freq define the colour of the LED and the frequency at which the LED is blinking. The following constants are defined to specify the colour: black, green, red, and yellow. When the colour black is specified the LED is turned off.

The constants no-blinking, slow, moderate, and fast are available to specify a frequency. The constant no-blinking causes the LED to be turned on permanently.

The following example shows how to let the second user LED blink at about 2 Hz (moderate) in red

ok red moderate 1 led!
ok

led-on (led# — ) turns the user LED identified by led# on. The value of led# may be either zero or one. The value zero specifies the first user LED, and the value one specifies the second user LED. The command only considers the state of the bit 0 of the value
led#. 

led-off ( led# — ) turns the user LED identified by led# off. The value of led# may be either zero or one. The value zero specifies the first user LED, and the value one specifies the second user LED. The command only considers the state of the bit 0 of the value led#.

led? ( led# — true | false ) determines the state of the LED identified by led#, and returns either true or false to indicate if the LED is turned on or off. The value of led# may be either zero or one. The value zero specifies the first user LED, and the value one specifies the second user LED. The command only considers the state of the bit 0 of the value led#.

When the LED is turned on, then the value true is returned; otherwise the value false is returned.

toggle-led ( led# — ) determines the state of the user LED identified by led#, and turns the LED on or off. The LED is turned on when it was turned off before, and vice versa. The value of led# may be either zero or one. The value zero specifies the first user LED, and the value one specifies the second user LED. The command only considers the state of the bit 0 of the value led#.

rotary-switch@ ( — byte ) returns the current state of the rotary switch. The value of byte may be one of the values in the range zero through 15. The value zero corresponds to the position 0 of the rotary switch, the value one corresponds to position 1, and so forth.

4.4.6 Reset

The command listed below are available to initiate various RESETs, and to obtain information about a previous RESET.

vme-sysreset ( — ) asserts the VMEbus SYSRESET* signal and thus causes a system reset.

reset-call ( — ) forces a local reset. This command provides the same function as the OpenBoot command reset.

vme-sysreset-in! ( true | false ) allows or prevents the board from being reset by the assertion of the VMEbus SYSRESET* signal. When the value true is passed to the command the board will be reset whenever the VMEbus SYSRESET* signal is asserted. Otherwise — the value false is passed to the command — the board will not be reset by the assertion of the SYSRESET* signal.
sbus-reset? ( — true | false ) determines whether the last reset occurred was due to an SBus reset. The value true is returned when the last reset was because of an SBus reset. Otherwise it returns the value false to indicate that the last reset was not because of an SBus reset.

wdt-reset? ( — true | false ) determines whether a reset has been generated because the watchdog timer has expired. If a reset has been generated because the watchdog timer reached the timeout value, then the value true is returned; otherwise the value false is returned.

vme-sysreset? ( — true | false ) determines whether the last reset occurred was due to the assertion of the VMEbus SYSRESET* signal. The value true is returned when the last reset was a VMEbus SYSRESET* reset. Otherwise it returns the value false to indicate that the last reset was not a VMEbus SYSRESET* reset.

vme-sysreset-call? ( — true | false ) determines whether the last reset occurred was due to a VMEbus SYSRESET* call. The value true is returned when the last reset was because of a VMEbus SYSRESET* call. Otherwise it returns the value false to indicate that the last reset was not a VMEbus SYSRESET* call.
A VMEbus SYSRESET* call is done by clearing the SYSRESET bit in the SPARC FGA-5000’s Miscellaneous Control and Status Register.

reset-call? ( — true | false ) determines whether the last reset occurred was due to a local reset call. The value true is returned when the last reset was because of a local reset call. Otherwise it returns the value false to indicate that the last reset was not a local reset call.
A local reset call is done by clearing the RESET bit in the SPARC FGA-5000’s Miscellaneous Control and Status Register.

vme-reset-call? ( — true | false ) determines whether the last reset occurred was due to a reset call initiated by an access via the VMEbus. The value true is returned when the last reset was because of a reset call. Otherwise it returns the value false to indicate that the last reset was not because of a reset call initiated by an access via the VMEbus.
A reset call is done by clearing the LOCRESET bit in the SPARC FGA-5000’s Global Control and Status Register.
4.5 Flash Memory Support

4.5.1 Flash Memory Programming

The commands listed below are available to access and program the flash memories available on the SPARC CPU-5V.

`flash-messages ( - vaddr )` returns the virtual address of the variable `flash-messages`. The state of this variable controls whether the words to erase and program the flash memories will display messages while erasing or programming the flash memories. Messages will not be displayed after `turning off` this variable by `flash-messages off`, and are displayed after `turning on` this variable by `flash-messages on`.

`flash-va ( - vaddr )` returns the virtual base address `vaddr` of the flash memory programming window. The virtual address returned is only valid when the flash memories have been previously prepared for accessing using the `select-flash` word.

`boot-flash-va ( - vaddr )` returns the virtual base address `vaddr` of the BOOT flash memory.

`user-flash-va ( - vaddr )` returns the virtual base address `vaddr` of the USER flash memory. When the USER flash memory is not accessible directly, but only through the flash memory programming window, then the address returned is zero. On the SPARC CPU-5V the USER flash memory is accessible only through the flash memory programming window. Thus, the commands described above have to be used to access the USER flash memory.

`select-flash ( "USER<eol>" | "BOOT<eol>" - )` prepares either the BOOT flash memories, or the USER flash memories for programming. In detail, the number and size of the available flash memories are determined, as well as the size of the flash memory programming window. The flash memory programming window is mapped and the virtual base address of the window is stored internally, and may be obtained by using the word `flash-va`.

`user-flash? ( - true | false )` checks whether the BOOT flash memory or the USER flash memory is accessible through the flash memory programming window. It returns `true` in the case that the USER flash memory is accessible through the programming window; otherwise it returns `false`.

`move>flash ( source-addr dest-addr count - )` programs the selected flash memory beginning at `dest-addr` with a number of bytes, specified by `count`, stored at `source-addr`.

`flash>move ( source-addr dest-addr count - )` copies a number of bytes, specified by `count`, from the selected flash memory beginning at `source-addr` to `dest-addr`. The flash memory is accessed through the flash memory programming window for reading data from the memory. Thus, the flash memory has to be prepared for accessing using the
command select-flash.

fill-flash (dest-addr count pattern — ) fills the selected flash memory beginning at dest-addr with a particular pattern. The number of bytes to be programmed in the flash memory is given by count.

erase-flash (device-number — ) erases a flash memory device identified by its device-number. The devices are numbered beginning from zero (0).

c!-flash (byte addr — ) stores the byte at the location within the selected flash memory identified by addr.

w!-flash (half-word addr — ) stores the half-word (16 bits) at the location within the selected flash memory identified by addr.

1!-flash (word addr — ) stores the word (32 bits) at the location within the selected flash memory identified by addr.

The USER flash memory is prepared for programming by:

ok select-flash USER
USER flash memory is selected for programming
Flash memory programming window at $ffe98000 size 512 Kbyte
512 Kbyte BOOT flash memory is available at $ffe58000.
2048 Kbyte USER flash memory is available.
ok

As shown above, the word select-flash informs the user that the USER flash memory has been made accessible through the flash memory programming window. It displays the base address (virtual address) of the window and its size.
The total amount of the available BOOT flash memory and USER flash memory is displayed, too. After the USER flash memory has been prepared for programming, all commands described above operate on the USER flash memory. And the BOOT flash memory is only read and programmed by these commands when the BOOT flash memory has been prepared for these operations by:

ok select-flash BOOT
BOOT flash memory is selected for programming
Flash memory programming window at $ffe98000 size 512 Kbyte
512 Kbyte BOOT flash memory is available at $ffe58000.
2048 Kbyte USER flash memory is available.
ok

To read data from the selected flash memory — in the current context from the USER flash memory — the command flash>move is used as follows:

ok flash-va h# 10.0000 h# 20.0000 flash>move
ok
The contents of the entire USER flash memory is copied to main memory beginning at address $10.0000_{16}$. A specific area within the selected flash memory is read by:

```
ok flash-va h# 6.8000 + h# 10.0000 h# 5.8c00 flash>move
ok
```

and copies 363520 bytes beginning from address flash-va + 6.8000$_{16}$ to main memory beginning at address $10.0000_{16}$.

### 4.5.2 Flash Memory Device

The device tree of OpenBoot for the SPARC CPU-5V contains a device node associated with the USER flash memories. Thus, it is possible to load an executable image stored in the available USER flash into memory and start such an executable.

The device is called "flash-memory@0,71300000" and is attached to the device node "/obio". The device alias flash is available as an abbreviated representation of the flash memory device path. The vocabulary of the flash memory device includes the standard commands recommended for a byte device. The words of this vocabulary are only available when the flash memory device has been selected as shown below:

```
ok cd flash
ok words
  close open selftest reset load
  write-blocks read-blocks seek write read
  max-transfer block-size
ok selftest .
0
ok device-end
ok
```

The example listed above, selects the flash memory device and makes it the current node. The word words displays the names of the methods of the VMEbus device. And the third command calls the method selftest and the value returned by this method is displayed. The last command unselects the current device node, leaving no node selected.

When the command select-dev is used to select the flash memory device, the NVRAM configuration parameters bootflash-#megs and bootflash-#devices have to be set properly, before the device can be selected.

The NVRAM configuration parameters listed below are available to control the loading of an image from the USER flash memory. The current state of these configuration parameters is displayed using the printenv command, and is modified using either the setenv, or the set-default command provided by OpenBoot.

bootflash-#megs specifies the amount of available USER flash memory in megabyte.
(default: 0 Megabyte)

`bootflash-#devices` specifies the number of available USER flash memory devices.
(default: no devices)

`bootflash-load-base` specifies the address where the data loaded from the available flash memory are stored when the `load` or `boot` command, provided by OpenBoot, is used to load an image from the flash memory.
When this parameter is set to -1 — which is the parameter’s default value — then the image loaded from the flash memory is stored beginning at the address `addr`. But when the value of the configuration parameter differs from -1, then the image loaded from the flash memory is stored beginning at the address specified by the configuration parameter `bootflash-load-base`. And the same address is stored in the variable `load-base` maintained by OpenBoot.

The methods listed below are available in the vocabulary of the flash memory device:

- `open` (— `true`) prepares the package for subsequent use. The value `true` is returned when the device has been opened successfully; otherwise the value `false` is returned. Usually, the fail state is indicated when the NVRAM configuration parameters `bootflash-#megs` and `bootflash-#devices` are not consistent.

- `close` (—) frees all resources allocated by open.

- `reset` (—) puts the flash memory device into quiet state.

- `selftest` (— `error-number`) always returns the value zero.

- `read` (`addr length` — `actual`) reads at most `length` bytes from the flash memory device into memory beginning at address `addr`. If `actual` is zero or negative, the read failed. The value of `length` may not always be a multiple of the device’s normal block size.

- `write` (`addr length` — `actual`) discards the information passed to the command and always returns zero to indicate that the device does not support this function.

- `seek` (`offset file#` — `error?`) seek to byte `offset` within the file identified by `file#`. The flash memory device package maintains an internal position counter that is updated whenever a method to read data from or to store data in the flash memories is called. If `offset` and `file#` are both zero, then the internal position counter is reset to offset zero, otherwise the value of `offset` is assigned to the internal position counter, and a subsequent access to the flash memories starts at the offset selected.
Because the flash memory device does not support any file system, the parameter `file#` is ignored, except in the case mentioned above.
When the seek succeeded the value of `error?` is zero, otherwise the value -1 is retun-
rnd to indicate the fail state.

`read-blocks ( addr block# #blocks — #read )` reads the number of blocks identified by
`#blocks` of length `block-size` bytes, each from the device beginning at the device
`bvlock block#`, into memory at address `addr`. It returns the number of blocks actually
read (`#read`).

`write-blocks ( addr block# #blocks — #written )` discards the information passed to the
command and always returns zero to indicate that the device does not support this
function.

`block-size (— bytes )` returns the size in bytes `bytes` of a block which is always the size of
the flash memory programming window.

`max-transfer (— bytes )` returns the size in bytes `bytes` of the largest single transfer the
device can perform. The command returns a multiple of `block-size`.

`load ( addr — length )` reads a stand-alone program from the flash memory beginning at off-
set `016` and stores it beginning at address `addr`. It returns the number of bytes `length`
read from the flash memory.

This method considers the state of the NVRAM configuration parameter `bootflash-load-base`: when this parameter is set to `-1` — which is the parameter’s
default value — then the image loaded from the flash memory is stored beginning at
the address `addr`. But when the value of the configuration parameter differs from `-1`,
then the image loaded from the flash memory is stored beginning at the address speci-
fied by the configuration parameter `bootflash-load-base`. And the same
address is stored in the variable `load-base` maintained by OpenBoot.

#### 4.5.3 Loading and Executing Programs from USER Flash Memory

Besides the ability to load and execute an executable image from disk, or via a network, or
other components, the OpenBoot for the SPARC CPU-5V provides a convenient way to load
and execute an executable image from the available USER flash memory. The executable
image to be loaded has to be either a binary image (a.out format), a FORTH program, or a
FCode program.

As mentioned at the beginning of this section the device alias `flash` is available as an
abbreviated representation of the flash memory device. The command listed below is used to
explicitly load and execute an image from the flash memory:

```
ok boot flash
```

The following NVRAM configuration parameters can be modified to determine whether or not
the system will load an executable image automatically after a power-up cycle or system reset:
auto-boot?
boot-device

Assuming, that the SPARC CPU-5V is equipped with one USER flash memory device which size is 1Mbyte, then commands listed in the following have to be used to load and execute an image from the flash memory automatically after a power-up cycle or system reset:

```
ok setenv bootflash-#devices 1
bootflash-#devices = 1
ok setenv bootflash-#megs 1
bootflash-#megs = 1
ok setenv boot-device flash
boot-device = flash
ok setenv auto-boot? true
auto-boot? = true
ok reset
```

4.5.4 Controlling the Flash Memory Interface

The commands listed below are available to control the flash memory interface. These commands are used to make a specific flash memory device available in the flash memory programming window, and to control the flash memory programming voltage.

flash-vpp-on (—) turns the programming voltage on.
flash-vpp-off (—) turns the programming voltage off.

userprom-select-page (page —) makes a page (one of a eight possible 512 KB pages) of a USER flash memory available in the flash memory programming window.

bootprom-select-page (page —) makes a page (one of a eight possible 512 KB pages) of a BOOT flash memory available in the flash memory programming window.

select-bootprom-1 (—) makes the first BOOT flash memory device available in the flash memory programming window.

select-bootprom-2 (—) makes the second BOOT flash memory device available in the flash memory programming window.

select-bootprom (device-number —) makes a BOOT flash memory device, identified by its device-number, available in the flash memory programming window. The devices are numbered beginning from zero (0).

select-userprom-1 (—) makes the first USER flash memory device available in the
flash memory programming window.

select-userprom-2 (-) makes the second USER flash memory device available in the
flash memory programming window.

select-userprom ( device — ) makes a USER flash memory device, identified by its
device-number, available in the flash memory programming window. The devices are
numbered beginning from zero (0).
4.6 Onboard Interrupts

Besides the interrupt handlers already available in the standard OpenBoot, the OpenBoot of the SPARC CPU-5V provides further handlers that deal with the interrupts generated by the following:

- one of the VMEbus interrupt levels one to seven;
- the assertion and negation of the SYSFAIL* signal;
- the assertion of the ACFAIL* signal;
- pressing the ABORT switch;
- the Watchdog Timer, when half the time has expired.

4.6.1 VMEbus Interrupts

The interrupt handlers for any VMEbus interrupt are not installed automatically by OpenBoot; however, appropriate words are available to activate and deactivate an interrupt handler serving a specific VMEbus interrupt. Such an interrupt handler is activated by:

```
ok 0 pil!
ok 3 5 install-vme-intr-handler
ok
```

The `pil!` command decreases the processor interrupt level to allow the processor to respond to all interrupts. By default, OpenBoot sets the mask to 13 and allows the processor to respond to interrupts above interrupt level 13. The second command installs the interrupt handler that deals with the VMEbus interrupt level 5. Furthermore, this command specifies that an SBus interrupt level 3 will be generated upon the occurrence of a VMEbus interrupt 5. Any of the seven SBus interrupt levels may be specified to be generated upon a VMEbus interrupt.

OpenBoot maintains seven variables called `vme-intr{1|2|3|4|5|6|7}-vector` which are modified by the VMEbus interrupt handlers. In general, the interrupt handlers store the vector obtained during an interrupt acknowledge cycle in the appropriate variable. The state of these variables is displayed by

```
ok .vme-vectors
ok
```

By default, the value -1 (true) is assigned to these variables to indicate that no VMEbus interrupt occurred. So, the word `.vme-vectors`, as shown above, will display "--" indicating that no interrupt occurred; otherwise it shows the vector obtained (a value in the range 0 to FF16).

Another way to display the state of a variable used to store the interrupt vector is

```
ok vme-intr5-vector ?
```
and the variable is set to -1 (true) by

ok vme-intr5-vector on
ok

An interrupt handler is removed and the corresponding interrupt is disabled by

ok 5 uninstall-vme-intr-handler
ok

All interrupt handlers serving all VMEbus interrupts are installed by

ok 0 pil!
ok 8 1 do i i install-vme-intr-handler loop
ok

In this case, all interrupt handlers are installed and the VMEbus interrupt to SBus interrupt mapping is as follows: SBus interrupt level 1 is generated upon the occurrence of a VMEbus interrupt 1; SBus interrupt level 2 is generated upon the occurrence of a VMEbus interrupt 2; and so forth.

4.6.2 SYSFAIL Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated upon the assertion and negation of the SYSFAIL* signal. This handler need not to be installed because it is already installed by OpenBoot.

By default, the interrupts that will be emitted by a status change of the SYSFAIL* signal are disabled and have to be enabled by

ok vme-sysfail-assert-mmiena
ok vme-sysfail-negate-mmiena
ok

which enable the generation of a nonmaskable interrupt whenever the SYSFAIL* signal is asserted and negated.

When a nonmaskable interrupt occurred due to the assertion of the SYSFAIL* signal, then the appropriate interrupt handler increments the variable sysfail-asserted? by one to report the occurrence of such an interrupt. The variable sysfail-negated? is incremented by the interrupt handler when the SYSFAIL* signal has been negated and caused a non-maskable interrupt. The state of both variables are obtained by

ok sysfail-asserted? ?
0
ok

and

ok sysfail-negated? ?
1
ok

And these variables are cleared — set to zero — by

ok sysfail-asserted? off
ok sysfail-negated? off
ok

4.6.3 ACFAIL Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated upon the assertion of the ACFAIL* signal. This handler need not to be installed because it is already installed by OpenBoot.

By default, the interrupt that will be emitted by asserting the ACFAIL* signal is disabled and has to be enabled by

ok vme-acfail-assert-irq-ena
ok

which enables the generation of a nonmaskable interrupt whenever the ACFAIL* signal is asserted.

When a nonmaskable interrupt occurred due to the assertion of the ACFAIL* signal, then the appropriate interrupt handler increments the variable acfail-asserted? by one to report the occurrence of such an interrupt. The state of this variable is obtained by

ok acfail-asserted? ?
2
ok

And the variable is cleared — set to zero — by

ok acfail-asserted? off
ok
4.6.4 ABORT Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated by pressing the front panel abort switch. This handler need not be installed because it is already installed by OpenBoot.

By default, the interrupt that will be emitted when the abort switch has been pressed is disabled and has to be enabled by

```
ok abort-nmi-ena
ok
```

which enables the generation of a nonmaskable interrupt whenever the abort switch is pressed.

When a nonmaskable interrupt occurred due to pressing the abort switch, then the appropriate interrupt handler increments the variable `abort-occurred?` by one to report the occurrence of such an interrupt. The state of both variables are obtained by

```
ok abort-occurred? ?
7
ok
```

And these variables are cleared — set to zero — by

```
ok abort-occurred? off
ok
```

Besides the effects described above, the pressing of the abort switch has the same effect as giving the `Stop-A` keyboard command. The program currently running is aborted and the FORTH interpreter appears immediately.

4.6.5 Watchdog Timer Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated by the watchdog timer when half of the time has expired. This handler need not to be installed because it is already installed by OpenBoot.

By default, the interrupt that will be emitted by the watchdog timer is disabled — the watchdog timer is disabled — and has to be enabled by

```
ok wd-nmi-ena
ok wd-ena
ok
```

In this example a nonmaskable interrupt is generated whenever half of the watchdog time has expired. The interrupt handler included in OpenBoot restarts the watchdog timer to ensure that the watchdog time will not expire and cause a reset. Additionally, the interrupt handler
increments the variable wdnmi-occurred? by one whenever the watchdog timer emits an interrupt. The state of this variable is determined by

```
ok wdnmi-occurred? ?
6
ok
```

This variable is cleared — set to zero — by

```
ok wdnmi-occurred? off
ok
```
SECTION 5

CIRCUIT SCHEMATICS

5. CPU-5V Schematics

Copies of the CPU-5V schematics are found on the next page. The schematics contain the signal and unit cross references as well as the history of the schematics.
<table>
<thead>
<tr>
<th>Value</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GND</td>
</tr>
<tr>
<td>1</td>
<td>SB_SEL&lt;1&gt;</td>
</tr>
<tr>
<td>2</td>
<td>SB_NR&lt;1&gt;</td>
</tr>
<tr>
<td>3</td>
<td>SB_D&lt;0&gt;</td>
</tr>
<tr>
<td>4</td>
<td>SB_INT&lt;1&gt;</td>
</tr>
<tr>
<td>5</td>
<td>SB_D&lt;2&gt;</td>
</tr>
<tr>
<td>6</td>
<td>SB_INT&lt;2&gt;</td>
</tr>
<tr>
<td>7</td>
<td>SB_D&lt;3&gt;</td>
</tr>
<tr>
<td>8</td>
<td>SB_INT&lt;3&gt;</td>
</tr>
<tr>
<td>9</td>
<td>SB_D&lt;4&gt;</td>
</tr>
<tr>
<td>10</td>
<td>SB_INT&lt;4&gt;</td>
</tr>
<tr>
<td>11</td>
<td>SB_D&lt;5&gt;</td>
</tr>
<tr>
<td>12</td>
<td>SB_INT&lt;5&gt;</td>
</tr>
<tr>
<td>13</td>
<td>SB_D&lt;6&gt;</td>
</tr>
<tr>
<td>14</td>
<td>SB_INT&lt;6&gt;</td>
</tr>
<tr>
<td>15</td>
<td>SB_D&lt;7&gt;</td>
</tr>
<tr>
<td>16</td>
<td>SB_INT&lt;7&gt;</td>
</tr>
<tr>
<td>17</td>
<td>SB_D&lt;8&gt;</td>
</tr>
<tr>
<td>18</td>
<td>SB_INT&lt;8&gt;</td>
</tr>
<tr>
<td>19</td>
<td>SB_D&lt;9&gt;</td>
</tr>
<tr>
<td>20</td>
<td>SB_INT&lt;9&gt;</td>
</tr>
<tr>
<td>21</td>
<td>SB_D&lt;10&gt;</td>
</tr>
<tr>
<td>22</td>
<td>SB_D&lt;11&gt;</td>
</tr>
<tr>
<td>23</td>
<td>SB_D&lt;12&gt;</td>
</tr>
<tr>
<td>24</td>
<td>SB_D&lt;13&gt;</td>
</tr>
<tr>
<td>25</td>
<td>SB_D&lt;14&gt;</td>
</tr>
<tr>
<td>26</td>
<td>SB_D&lt;15&gt;</td>
</tr>
<tr>
<td>27</td>
<td>SB_D&lt;16&gt;</td>
</tr>
<tr>
<td>28</td>
<td>SB_D&lt;17&gt;</td>
</tr>
<tr>
<td>29</td>
<td>SB_D&lt;18&gt;</td>
</tr>
<tr>
<td>30</td>
<td>SB_D&lt;19&gt;</td>
</tr>
<tr>
<td>31</td>
<td>SB_D&lt;20&gt;</td>
</tr>
<tr>
<td>32</td>
<td>SB_D&lt;21&gt;</td>
</tr>
<tr>
<td>33</td>
<td>SB_D&lt;22&gt;</td>
</tr>
<tr>
<td>34</td>
<td>SB_D&lt;23&gt;</td>
</tr>
<tr>
<td>35</td>
<td>SB_D&lt;24&gt;</td>
</tr>
<tr>
<td>36</td>
<td>SB_D&lt;25&gt;</td>
</tr>
<tr>
<td>37</td>
<td>SB_D&lt;26&gt;</td>
</tr>
<tr>
<td>38</td>
<td>SB_D&lt;27&gt;</td>
</tr>
<tr>
<td>39</td>
<td>SB_D&lt;28&gt;</td>
</tr>
<tr>
<td>40</td>
<td>SB_D&lt;29&gt;</td>
</tr>
<tr>
<td>41</td>
<td>SB_D&lt;30&gt;</td>
</tr>
<tr>
<td>42</td>
<td>SB_D&lt;31&gt;</td>
</tr>
<tr>
<td>43</td>
<td>SB_D&lt;32&gt;</td>
</tr>
<tr>
<td>44</td>
<td>SB_D&lt;33&gt;</td>
</tr>
<tr>
<td>45</td>
<td>SB_D&lt;34&gt;</td>
</tr>
<tr>
<td>46</td>
<td>SB_D&lt;35&gt;</td>
</tr>
<tr>
<td>47</td>
<td>SB_DP</td>
</tr>
</tbody>
</table>

**Notes:**
- **逻辑值**: Value
- **命名**: NAME
- **电源电压**: +12V
- **设计者**: SK
- **日期**: 1995年4月25日
- **图号**: SBUS_CONN
- **版本**: 0.2

**电路描述**:
- SBUS_SELECT
- SBUS_CONN 连接
- SBUS<YSBUS>
ONLY ONE OF THESE BLOCKS CAN BE ASSEMBLED

MIN_LINE_WIDTH=10

LOC_BUS<XLOC><LOCBS>
### PORT A:

<table>
<thead>
<tr>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXDA</td>
<td>3</td>
<td>TXDA</td>
<td>24</td>
<td>TXDA/</td>
<td>24</td>
<td>TXDA/</td>
<td>2</td>
</tr>
<tr>
<td>RTS</td>
<td>4</td>
<td>RTS</td>
<td>9</td>
<td>TXB/</td>
<td>14</td>
<td>TXB/</td>
<td>7</td>
</tr>
<tr>
<td>RTS</td>
<td>6</td>
<td>RTS</td>
<td>12</td>
<td>TXB/</td>
<td>14</td>
<td>TXB/</td>
<td>11</td>
</tr>
<tr>
<td>RTS</td>
<td>8</td>
<td>RTS</td>
<td>20</td>
<td>TXB/</td>
<td>14</td>
<td>TXB/</td>
<td>11</td>
</tr>
<tr>
<td>RTS</td>
<td>10</td>
<td>RTS</td>
<td>20</td>
<td>TXB/</td>
<td>14</td>
<td>TXB/</td>
<td>11</td>
</tr>
<tr>
<td>RTS</td>
<td>12</td>
<td>RTS</td>
<td>20</td>
<td>TXB/</td>
<td>14</td>
<td>TXB/</td>
<td>11</td>
</tr>
<tr>
<td>DSR</td>
<td>16</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
</tr>
<tr>
<td>DSR</td>
<td>18</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
</tr>
</tbody>
</table>

### PORT B:

<table>
<thead>
<tr>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
<th>SCC CONNECTOR</th>
<th>PIN #</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXDB</td>
<td>1</td>
<td>TXDB</td>
<td>14</td>
<td>RXDB</td>
<td>25</td>
<td>RXDB</td>
<td>22</td>
</tr>
<tr>
<td>TXD</td>
<td>2</td>
<td>TXD</td>
<td>6</td>
<td>TXD</td>
<td>6</td>
<td>TXD</td>
<td>6</td>
</tr>
<tr>
<td>RTS</td>
<td>4</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
</tr>
<tr>
<td>RTS</td>
<td>6</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
</tr>
<tr>
<td>RTS</td>
<td>8</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
</tr>
<tr>
<td>RTS</td>
<td>10</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
</tr>
<tr>
<td>RTS</td>
<td>12</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
<td>RTS</td>
<td>9</td>
</tr>
<tr>
<td>DSR</td>
<td>16</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
</tr>
<tr>
<td>DSR</td>
<td>18</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
<td>DSR</td>
<td>25</td>
</tr>
</tbody>
</table>

### ADAPT SERIAL

- **SER BUS<XSERO>1**
- **SER BUS<XSERO>1**

### VALUE

- **27K**
- **[MAX]**

### DESIGNER: SK

### DRAWING

- **ABBREV=SERIAL**
- **SHEET 23**
- **REV. 0.2**

**COMPUTERS**  **SERIAL**
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE = DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE = DRAM4M4R4)
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
HISTORY FOR SOURCE SCHEMATIC CHANGES

Changes from CPU-SV Revision 0.0 to Revision 0.1:
(File: ..../doc/changes/history_0_1)

Standard serial PROM connection implemented:
- serial PROM counter reset
- reset key applicable
Changes on Sh 18 J3.

Signal JTAG_RST, JTAG_THS, JTAG_CLK JTAG_TDI_TDO<4> pulled
up with 10k and pulled down with 33k (33k RX).
Changes on Sh 01 B4.

Signals EB_D<21, 20> driven by LCA for Flash PROM.
Changes on Sh 17 J38, SH 18 J58.

CAS<2> and CAS<3> swapped and CAS<0> and CAS<1> swapped.
Changes on Sh 1 J35.

PA<22> and PA<26> swapped for MAC10.
Changes on Sh 1 J52.

SP_SEL<1, 0> connected with the correct
pins of the bus connectors.
Changes on Sh 3 J34, SH 4 J41, SH 5 P6.

Position of LED 1 and 2 exchanged.
Changes on Sh 19 J43.

Fuse replaced with IC10.
Changes on Sh 8, SH 12, SH 25.

Signals EB_D<3, 0> connected with LCA instead of EB_D<7..4>.
Changes on Sh 18 J36.

DRAM Data Bus pulled.
Changes on Sh 18 J2.

SYSRESET in through LCA to FGA-5000.
Changes on Sh 18 J88, SH 30 J55.

SYSRESET out from LCA in addition.
Changes on Sh 18 J36, SH 31 J77.

SYSRESET out driven by 3 transistors.
Changes on Sh 18 J46, SH 29 J31, SH 30 J35, SH 31 J49.

Signals VSI.TEST, VSI.TESTASYNC and VSI.TESTSCAN pulled up and down.
Changes on Sh 30 J35.

Signals ACFAIL, SVFAIL, SYSRESET, EBSY, IRQ<7..0> glitch filtered.
Changes on Sh 31 J31, SH 32 J32, SH 33.

5 row VHEbus connector P0 pins disconnected.
Changes on Sh 18.

5 row VHEbus connector P1 only GND, D disconnected.
Changes on Sh 39.

3 row VHEbus connector P2, Centronics: D1..D19 shifted to D3..D21.
Changes on Sh 40.

Geographical address lines connected with CPUs.
Changes on Sh 31.

GND for serial interface connected with hybride GND.
Changes on Sh 24.

Additional GND footprint for Q6. VSI_CLK: QUOM Q8.
Changes on Sh 16.

Dislike UVBERRI and UVBTRYI when not being VHEbus master.
Changes on Sh 18, J30, J34.

Delay negative signal BERR* when not being VHEbus master.
Changes on Sh 18, J30, J34.

Additional block C8 installed.
Changes on Sh 18.

Voltage division resistors for 3.5V changed from 0805 to 1206.
Changes on Sh 30.

MIN_LINE_WIDTH add to VHEbus signals.
Changes on Sh 31.

Holes entered into schematics.
Changes on Sh 84.

Replace LT1085 with LT1085 ADJ.
Changes on Sh 30 J20, R96, R210.

RCS<0, 2, 4, 6> assertion delayed.
Changes on Sh 43.

Relais removed.
Changes on Sh 19 J76, D22, R164.

Quam 80 MHz is used only for FGA-5000, J18 (FACT74) removed.
Changes on Sh 43.

Clock signal for SC17 generated with new quartz Q9 instead of FACT4.
Changes on Sh 16.

Signal ENET.TCLK used for SLaudio signal TIM_CLK (10 mHz).
Changes on Sh 19.

MIN_LINE_WIDTH of quartz signals changed from 0 mil to 10 mil.
Changes on Sh 16.

Signals ENET.TCLK and ENET.RCLK terminated on the destination end.
Changes on Sh 11, 16, 83.

Changes from CPU-SV Revision 0.1 to Revision 0.2:
(File: ..../doc/changes/history_0_2)

Bus AITA<3, 0> added.
Changes on Sh 1.

History sheet number updated.
Changes on Sh 1.

Serial PROM JC1736 replaced with XCl765.
Changes on Sh 18 J31.

SYSFAIL connected with LCA.
Changes on Sh 19 J36.

Parts added in order to support third overtone crystals.
Changes on Sh 17 Kx, Kx, Kx, Kx, INDX.

DRAM_CODE A<> connected.
Changes on Sh 29 _ADAPT_DRAM_CODE.

AMUSB controlled by AMUSB, UPACK and VHADIR.
Changes on Sh 15 _ADAPT_AMUSB, SH 32 J93.

History updated.
Changes on Sh 93.

DESIGNER: SK
Thu Mar 23 00:09:42 1995
Apr/25/95 Manual

COlUMN A

FORMAT

COlUMNS D

HISTORY REV.

COlUMN C

1 2 3 4

A B C D
Changes from CPU-5V Revision 0.1 to Revision 0.2:
(File: ../../doc/changes/history_0_2)

Bus AMIA+3 02 added.
Changes on SH 1.
History sheet number updated.
Changes on SH 1.
Serial FROM XC1765 replaced with XC1765.
Changes on SH 15 041.
SYSFAIL0 connected with LCA.
Changes on SH 15 06.
Parts added in order to support third overtone crystals.
Q1, R321, R322, C34, IN08.
DRAM_CODE_A 5 added.
Changes on SH 29 _ADAPT_DRAM_CODE.
ANVSEL controlled by AM3VME, VJACK and VMADIR.
Changes on SH 35 _ADAPT_AMIA, SH 42 053.
SYSRESET is prohibited with an additional transistor and
resistor when the board is not a slot 1 device.
Changes on SH 32 06, R07.
Keyboard and mouse are connected with VMEbus P2 by default.
Changes on SH 38 R240 R050 R253.
Resistors for JTAG changed.
Changes on SH 32 R14 R28.
JTAG delay chain closed.
Changes on SH 1.
Resistor value changed for RAS timing.
Changes on SH 32 R20.
Capacitor removed.
Changes on SH 31 0113.
Socket for LFC changed.
Changes on SH 28 051.
Sockets for GALs added.
Changes on SH 2 J6, SH 3 J56.
PCB changed to Rev. 0.2.
Changes on SH 85 0F04.
Stiffener body changed.
Changes on SH 85 0511.
Parameter name changed: SR_SELECT instead of SR SLOT.
Changes on SH 4 0 0 5, SH 6 0 9, SH 8.
Switch SW 4-4 is connected with Select GAL.
Changes on SH 2 J6, R322, R213.
Switch SW 5-4 is connected with LCA.
Changes on SH 19 06, R08, R08.
History updated.
Changes on SH 94.

HISTORY FOR SOURCE SCHEMATIC CHANGES

DESIGNER: SK
Mon Apr 3 08:12:55 1995

DRAMING: APR/25/95 Manual
ABBREV-HISTORY: COMPUTERS
SHEET: 94
OF: 94
REV.: 0.2
5.1 MEM-5 Schematics
60 NS DRAMS EINTRAGEN!!!
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR=(CHIP_TYPE=DRAM4M4R4)

LOCATION=J26
$LOCATION=329

<table>
<thead>
<tr>
<th>B_MEM_A&lt;11&gt;</th>
<th>B_MEM_A&lt;10&gt;</th>
<th>D&lt;63&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_MEM_A&lt;9&gt;</td>
<td>B_MEM_A&lt;8&gt;</td>
<td>D&lt;62&gt;</td>
</tr>
<tr>
<td>B_MEM_A&lt;7&gt;</td>
<td>B_MEM_A&lt;6&gt;</td>
<td>D&lt;61&gt;</td>
</tr>
<tr>
<td>B_MEM_A&lt;5&gt;</td>
<td>B_MEM_A&lt;4&gt;</td>
<td>D&lt;60&gt;</td>
</tr>
<tr>
<td>B_MEM_A&lt;2&gt;</td>
<td>B_MEM_A&lt;1&gt;</td>
<td></td>
</tr>
<tr>
<td>B_MEM_A&lt;0&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B_MEM_A&lt;10&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B_MEM_A&lt;3&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TSOP**

**DRAMSPEZ2**

$EXPR=(CHIP\_TYPE=DRAM4M4R4)$
$LOCATION=J21

TSOP

DRAMSPEZ2

EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR=(CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)
EXPR = (CHIP_TYPE=DRAM4M4R4)

LOCATION=J38

$LOCATION=J38

\begin{align*}
B_{MEM} A_{11} & \\
B_{MEM} A_{10} & \\
B_{MEM} A_{9} & \\
B_{MEM} A_{8} & \\
B_{MEM} A_{7} & \\
B_{MEM} A_{6} & \\
B_{MEM} A_{5} & \\
B_{MEM} A_{4} & \\
B_{MEM} A_{3} & \\
B_{MEM} A_{2} & \\
B_{MEM} A_{1} & \\
B_{MEM} A_{0} & \\
B_{MEM} A_{10} & \\
\end{align*}
SECTION 6

6. Insert your OPEN BOOT 2.0 PROM MANUAL SET here.
7. Product Error Report

Dear Customer,

Although FORCE COMPUTERS has achieved a very high standard of quality in products and documentation, suggestions for improvements are always welcome.

Customer feedback is always appreciated.

Please use the “Product Error Report” form on the next page for your comments and return it to one of our listed offices.

Sincerely,

FORCE COMPUTERS GmbH/Inc.
This page intentionally left blank
# PRODUCT ERROR REPORT

## HARDWARE/SOFTWARE/SYSTEMS

<table>
<thead>
<tr>
<th>PRODUCT:</th>
<th>SERIAL NO.:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE OF PURCHASE:</td>
<td>ORIGINATOR:</td>
</tr>
<tr>
<td>COMPANY:</td>
<td>POINT OF CONTACT:</td>
</tr>
<tr>
<td>ADDRESS:</td>
<td>TELEPHONE:</td>
</tr>
<tr>
<td></td>
<td>EXT:</td>
</tr>
</tbody>
</table>

**PRESENT DATE:**

**THIS AREA TO BE COMPLETED BY FORCE COMPUTERS:**

<table>
<thead>
<tr>
<th>DATE:</th>
<th>PR#:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSIBLE DEPT.:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENGINEERING</td>
</tr>
<tr>
<td></td>
<td>MARKETING</td>
</tr>
<tr>
<td></td>
<td>PRODUCTION</td>
</tr>
</tbody>
</table>

**AFFECTED PRODUCT:**

<table>
<thead>
<tr>
<th>AFFECTED DOCUMENTATION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARDWARE</td>
</tr>
<tr>
<td>SOFTWARE</td>
</tr>
<tr>
<td>SYSTEM</td>
</tr>
</tbody>
</table>

**ERROR DESCRIPTION:**

Please send this product error report to one of our nearest FORCE COMPUTERS offices:

**FORCE COMPUTERS Inc.**  
2001 Logic Drive  
San Jose, CA 95124-3468  
U.S.A.

**FORCE COMPUTERS GmbH**  
Prof.-Messerschmitt-Str.1  
D-85579 Neubiberg/Munich  
Germany

**FORCE COMPUTERS France S.A.R.L.**  
Le Volta  
17-19 rue Jeanne Bracconier  
F-92366 Meudon La Forêt Cédex  
France

**FORCE COMPUTERS U.K. Ltd.**  
Alton House Office Park  
Gatehouse Way  
Aylesbury, Bucks. HP19 3XU  
United Kingdom