MasPar MP-1 Hardware Manuals

Installation and Service Manual
Architecture Specification
System Administration
Release Notes

Part Number 9300-9022-00, Revision A3
MasPar Computer Corporation
MasPar MP-1 Service Manual Part Number 9300-9023, Revision A1, July 1992

MasPar Computer Corporation makes no representation or warranties regarding the contents of this document and reserves the right to revise this document or make changes in the specifications of the product described herein at any time without obligation to notify any person of such revisions or changes.

Copyright © 1992 by MasPar Computer Corporation, Sunnyvale, CA
All Rights Reserved.
Printed in the United States of America

This manual, its format, and the hardware, software, and microcode described in it are copyrighted by MasPar Computer Corporation and may not be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the express written permission of MasPar Computer Corporation.

Restricted Rights Legend

The software and microcode documented in this manual has been developed at private expense and is not in the public domain. Use, duplication, or disclosure by the U.S.Government is subject to restrictions as set forth in subdivision (c) (l) (i) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

FCC Notice

Note: This equipment has been tested and found compliant with the limits for a CLASS A digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

MasPar and the MasPar logo are registered trademarks of MasPar Computer Corporation.

UNIX is a registered trademark of the American Telephone and Telegraph Company.

DEC, DECstation, and ULTRIX are trademarks of Digital Equipment Corporation.

MasPar Computer Corporation
749 North Mary Avenue
Sunnyvale, California, 94086
408-736-3300
408-736-9560 (FAX)
Preface

This manual provides installation, service, and diagnostic procedures for MasPar® MP-1 systems and descriptions of the system components. Anyone who installs or services the MP-1 system or needs to replace any items should read this manual and be familiar with the procedures.

Refer to the Digital Equipment Corporation (DEC) manuals for instructions regarding the DECstation 5000 and service support.
How This Manual is Organized

Chapter 1 has system block diagrams and a general overview for both the data-parallel unit (DPU) and the DECstation.

Chapter 2 describes how to unpack the MP-1 system and install the system components and cables.

Chapter 3 describes the system switches, controls, DPU power system settings and adjustments, and how to power the system up and down.

Chapter 4 describes the LED indicators, error logs, and diagnostic software provided by MasPar. It tells how to use them, and how to interpret the results.

Chapter 5 provides the procedures for removing and replacing the DPU components.

Chapter 6 provides the procedures for upgrading the MP-1 system by adding more processor element (PE) array boards.

Appendix A is a brief installation checklist.

Appendix B provides instructions for preparing systems outside the United States to conform to local wiring standards.
Conventions Used in this Manual

Note Definitions

Three types of notes are used in this manual:

**NOTE:** gives additional information or information particularly important to the procedure.

**CAUTION:** indicates potential damage to equipment or data.

**WARNING:** indicates potential injury to the user.

Notation Conventions

typewriter  
Examples of input or output appear in typewriter font. Enter text exactly as shown.

*italics*  
Parameters provided by users are shown in *italics*.

[]  
Optional elements are enclosed in square brackets.
Related Publications

The following publications provide additional information about the MP-1 system.


DECstation 5000 Hardware Installation Guide, DEC #EK-365AA-IN-002

MasPar System Administration, PN 9300-0600

MasPar Architecture Specification, PN 9300-5001

MasPar Parallel VME Manual, PN 9300-9018

MasPar I/O Channel Controller and MasPar I/O RAM PCB Service Manual, PN 9300-9043

MasPar Site Prep Guide, PN 9300-5000

MasPar 3000-Series Parallel Disk Array Manual, PN 9300-5005

MasPar 4000-Series Parallel Disk Array Manual, PN 9300-9040
Contacting MasPar

We try to write manuals that will help you use MasPar products effectively and efficiently. Your comments play an important part in helping us do this. To make comments or report defects on MasPar products and documentation, you can contact MasPar via email, phone, or FAX at these locations:

**In North America and the Pacific Rim:**
Customer Support
MasPar Computer Corporation
749 North Mary Avenue
Sunnyvale
California
94086

*phone: 1-800-526-0916  
hours: 8AM - 6PM PST

*FAX: 408-736-9560  
email: support@maspar.com

**In Europe and the United Kingdom:**
European Customer Support
MasPar Computer Ltd
8 Commerce Park
Brunel Road
Theale
Reading RG7 4AB ENGLAND

*phone: +44-734-305444  
hours: 9AM - 5PM BST

*FAX: +44-734-305505  
email: support@mpread.co.uk
vi Preface
# Table of Contents

**Preface**

- How This Manual is Organized ........................................... ii
- Conventions Used in this Manual ...................................... iii
  - Note Definitions ....................................................... iii
  - Notation Conventions ................................................ iii
- Related Publications .................................................... iv
- Contacting MasPar ....................................................... v

**Chapter 1.**

**MasPar System Overview** ........................................... 1-1

- Models ............................................................................. 1-3
- Front-End ........................................................................ 1-5

**Data Parallel Unit** ...................................................... 1-6
  - DPU Card Cage .......................................................... 1-8
  - Parallel VME (PVME) PCB ........................................... 1-10
    - I/O RAM Buffer ...................................................... 1-10
    - VME Features ....................................................... 1-10
  - I/O Channel Controller (IOCTLR) PCB and I/O RAM (IORAM) PCB ........................................... 1-11
  - MVIB PCB ................................................................ 1-11
  - The Array Control Unit (ACU) ...................................... 1-12
  - PE PCB and Router PCB .............................................. 1-12
  - PE Array ..................................................................... 1-13
  - PE Numbering ............................................................ 1-15
  - PE Communications .................................................... 1-16
    - ACU-PE Bus ............................................................ 1-16
    - X-Net and Global Router Communications ...................... 1-16
  - X-Net ......................................................................... 1-18
  - Global Router ............................................................ 1-19
Chapter 2. System Hardware Installation

- Initial Inspection ........................................... 2-2
- Unpacking the DPU ........................................... 2-3
- Unpacking the DEC DECstation 5000 ....................... 2-8
- DECstation Installation ...................................... 2-8
- DPU Installation ............................................ 2-9
  - Handling PCBs Correctly ................................ 2-9
  - DPU PCBs .................................................. 2-9
  - ACU Jumper Settings ................................... 2-10
    - ACU Jumper Synopsis ................................ 2-11
  - DPU Card Cage Access ................................... 2-13
  - DPU Card Cage Slots ................................... 2-14
  - Installing the PCBs .................................... 2-15
- DPU Backplane Jumpers ..................................... 2-16
- Connecting the DPU and DECstation Cables ............... 2-18
  - Connecting the Internal DPU Cables .................. 2-18
  - Connecting the External DPU Cables .................. 2-19
  - System Power Requirements ............................ 2-19
- Preparing the System Modem ............................... 2-20
- Connecting the DPU Power Cable ......................... 2-21
  - Power Supply Levels .................................. 2-21

Chapter 3. Power and Controls

- Controls Overview ......................................... 3-2
- DPU Power System ......................................... 3-3
- Normal Operation ........................................... 3-5
- Adjusting DPU Power Supply ............................... 3-6
- Powering the System Up and Down ......................... 3-11
  - Power-Up Sequence .................................... 3-11
  - Power-Down Sequence .................................. 3-12

Chapter 4. Troubleshooting

- DPU LEDs .................................................. 4-2
  - DPU Front Panel LEDs .................................. 4-2
  - Printed Circuit Board LEDs ............................ 4-4
  - Power Supply LEDs ..................................... 4-7
- Interpreting Log File Messages ........................... 4-8
  - /usr/adm/dpujobmgr.log ................................ 4-8
  - PE Numbering ........................................... 4-9
Chapter 5. Field Replacement Procedures

Card Cage Access .................................................. 5-2

DPU Card Cage Slots ............................................. 5-3
I/O Slots ................................................................. 5-4
PE Array Board Slots ............................................. 5-4

Replacing DPU Card Cage PCBs ............................... 5-5
  Replacing the ACU ................................................. 5-6
  ACU Jumper Settings ........................................... 5-7
  ACU Jumper Synopsis .......................................... 5-7
  Replacing the MVIB PCB ....................................... 5-9
  Replacing PE Array and Router PCBs ....................... 5-10

Replacing DPU Power Trays .................................... 5-11
  MP 1200 Power Tray ........................................... 5-11
    Removal ......................................................... 5-11
    Replacement ................................................... 5-12
  MP 1100 Power Tray ........................................... 5-14
    Removal ......................................................... 5-14
    Replacement ................................................... 5-16

Replacing DPU Fan Tray ........................................ 5-18
  MP 1200 Fan Tray ............................................... 5-18
  MP 1100 Fan Tray ............................................... 5-19

Chapter 6. Upgrading PE Arrays ............................... 6-1

System Issues for Upgrades ................................... 6-2
Handling PCBs ...................................................... 6-3
Adding PE Array PCBs .......................................... 6-4

DPU Backplane Jumpers ......................................... 6-7
  Backplane Access ............................................... 6-9
  Xnet Jumpers ................................................... 6-10

Appendix A. Installation Checklist .......................... A-1

Appendix B. Power Cable Wiring and Replacement ........ B-1

Power Cable Wires ................................................ B-1
  Wire Ends ......................................................... B-1

Replacing the Power Cable .................................... B-2
Index
## List of Figures

<p>| Figure 1-1 | System Block Diagram | 1-2 |
| Figure 1-2 | Typical MP-1 Installation | 1-4 |
| Figure 1-3 | MasPar System Diagram--DECstation 5000 | 1-5 |
| Figure 1-4 | MasPar System Diagram--DPU with PVME | 1-7 |
| Figure 1-5 | MasPar System Diagram--DPU with IOCTLR and an IORAM | 1-7 |
| Figure 1-6 | DPU Card Cages | 1-9 |
| Figure 1-7 | PE Block Diagram | 1-13 |
| Figure 1-8 | The PE Array | 1-14 |
| Figure 1-9 | System Configuration Codes | 1-15 |
| Figure 1-10 | X-Net and Global Router Communications | 1-17 |
| Figure 1-11 | The Eight Directions on the X-Net | 1-18 |
| Figure 1-12 | Global Router Layout in a Model 1202 System | 1-20 |
| Figure 2-1 | Unopened Shipping Container | 2-4 |
| Figure 2-2 | Removing the Front Board | 2-5 |
| Figure 2-3 | Ramp Ready to Use | 2-6 |
| Figure 2-4 | Setting the Levelers | 2-7 |
| Figure 2-5 | PCB Ejector Levers | 2-10 |
| Figure 2-6 | ACU Jumpers | 2-12 |
| Figure 2-7 | Back of MP 1200 System and Card Cage | 2-13 |
| Figure 2-8 | DPU Card Cage Slots | 2-14 |
| Figure 2-9 | DPU Backplane Jumpers | 2-17 |</p>
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-10</td>
<td>DPU-DECstation Cables</td>
<td>2-18</td>
</tr>
<tr>
<td>2-11</td>
<td>DECstation Rear Panel Connectors</td>
<td>2-19</td>
</tr>
<tr>
<td>3-1</td>
<td>Controls (Front of DPU)</td>
<td>3-1</td>
</tr>
<tr>
<td>3-2</td>
<td>Controls (Rear of DPU)</td>
<td>3-2</td>
</tr>
<tr>
<td>3-3</td>
<td>DPU Backplane Voltage Test Points</td>
<td>3-7</td>
</tr>
<tr>
<td>3-4</td>
<td>MP 1100 Power Supply Voltage Adjustments</td>
<td>3-9</td>
</tr>
<tr>
<td>3-5</td>
<td>MP 1200 Power Supply Wiring and Voltage Adjustments</td>
<td>3-10</td>
</tr>
<tr>
<td>3-6</td>
<td>DECstation Rear Panel Connectors</td>
<td>3-11</td>
</tr>
<tr>
<td>4-1</td>
<td>DPU Front Panel LEDs</td>
<td>4-2</td>
</tr>
<tr>
<td>4-2</td>
<td>Printed Circuit Board LEDs</td>
<td>4-4</td>
</tr>
<tr>
<td>4-3</td>
<td>Power Supply LEDs</td>
<td>4-7</td>
</tr>
<tr>
<td>4-4</td>
<td>System Configuration Codes</td>
<td>4-9</td>
</tr>
<tr>
<td>5-1</td>
<td>Back of MP 1200 System and Card Cage</td>
<td>5-2</td>
</tr>
<tr>
<td>5-2</td>
<td>DPU Card Cage Slots</td>
<td>5-3</td>
</tr>
<tr>
<td>5-3</td>
<td>PCB Ejector Levers</td>
<td>5-5</td>
</tr>
<tr>
<td>5-4</td>
<td>ACU Jumpers</td>
<td>5-8</td>
</tr>
<tr>
<td>5-5</td>
<td>Power Tray Rear Panel</td>
<td>5-13</td>
</tr>
<tr>
<td>5-6</td>
<td>MP 1100 Power Supplies</td>
<td>5-15</td>
</tr>
<tr>
<td>6-1</td>
<td>PCB Ejector Levers</td>
<td>6-3</td>
</tr>
<tr>
<td>6-2</td>
<td>DPU Rear Doors</td>
<td>6-5</td>
</tr>
<tr>
<td>6-3</td>
<td>DPU Card Cage Slots</td>
<td>6-6</td>
</tr>
<tr>
<td>6-4</td>
<td>DPU Backplane Jumpers</td>
<td>6-8</td>
</tr>
<tr>
<td>6-5</td>
<td>System Front Doors</td>
<td>6-9</td>
</tr>
<tr>
<td>6-6</td>
<td>MP 1100 Xnet Jumper Configurations</td>
<td>6-10</td>
</tr>
<tr>
<td>6-7</td>
<td>MP 1200 Xnet Jumper Configurations</td>
<td>6-11</td>
</tr>
<tr>
<td>B-1</td>
<td>Power Cable Wiring</td>
<td>B-4</td>
</tr>
<tr>
<td>Table 1-1</td>
<td>MP-1 System Models .................................................. 1-3</td>
<td></td>
</tr>
<tr>
<td>Table 3-1</td>
<td>DPU Power Settings ..................................................... 3-5</td>
<td></td>
</tr>
<tr>
<td>Table 4-1</td>
<td>Front Panel LEDs ......................................................... 4-3</td>
<td></td>
</tr>
<tr>
<td>Table 4-2</td>
<td>ACU LEDs ................................................................. 4-5</td>
<td></td>
</tr>
<tr>
<td>Table 4-3</td>
<td>PE and Router LEDs .................................................... 4-6</td>
<td></td>
</tr>
<tr>
<td>Table 4-4</td>
<td>FLTCOD Values .......................................................... 4-10</td>
<td></td>
</tr>
<tr>
<td>Table 4-5</td>
<td>PMSTAT Values .......................................................... 4-11</td>
<td></td>
</tr>
<tr>
<td>Table 4-6</td>
<td>Diagnostic Tests, Sheet 1 of 2 Sheets ............................. 4-18</td>
<td></td>
</tr>
<tr>
<td>Table 4-7</td>
<td>Diagnostic Tests, Sheet 2 of 2 Sheets ............................. 4-19</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

MasPar System Overview

The MasPar® MP-1 parallel processing system is a massively parallel, data-parallel processing system. Each model has at least 1,024 simple, parallel, data processor elements. The MasPar system consists of two boxes: a front-end and a DPU. Optionally, a disk array subsystem in its own cabinet is available.

This chapter provides a brief description of the MasPar system. Figure 1-1 shows the general structure of the MasPar system.
Figure 1-1  System Block Diagram
Models

The MP-1 includes MP 1100 and MP 1200 Series systems. MP 1100 Series systems support 1, 2, or 4 PE array boards and have 5 I/O slots. MP 1200 Series systems support 1, 2, 4, 8, or 16 PE array boards and have 15 I/O slots. A PE board contains 1024 4-bit processor elements.

The model number is derived from the series number and the number of PE array boards, as shown in Table 1-1.

<table>
<thead>
<tr>
<th>Model</th>
<th>Enclosure Type</th>
<th>Processor Elements</th>
<th>Memory (MByte)</th>
<th>I/O Slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP 1101C/D</td>
<td>1100</td>
<td>1,024</td>
<td>16/64*</td>
<td>5</td>
</tr>
<tr>
<td>MP 1102C/D</td>
<td>1100</td>
<td>2,048</td>
<td>32/128*</td>
<td>5</td>
</tr>
<tr>
<td>MP 1104C/D</td>
<td>1100</td>
<td>4,096</td>
<td>64/256*</td>
<td>5</td>
</tr>
<tr>
<td>MP 1201E/F</td>
<td>1200</td>
<td>1,024</td>
<td>16/64**</td>
<td>15</td>
</tr>
<tr>
<td>MP 1202E/F</td>
<td>1200</td>
<td>2,048</td>
<td>32/128**</td>
<td>15</td>
</tr>
<tr>
<td>MP 1204E/F</td>
<td>1200</td>
<td>4,096</td>
<td>64/256**</td>
<td>15</td>
</tr>
<tr>
<td>MP 1208E/F</td>
<td>1200</td>
<td>8,192</td>
<td>128/512**</td>
<td>15</td>
</tr>
<tr>
<td>MP 1216E/F</td>
<td>1200</td>
<td>16,384</td>
<td>256/1,024**</td>
<td>15</td>
</tr>
</tbody>
</table>

* MP 1100-Series "D" models have larger memory capacity.
** MP 1200-Series "F" models have larger memory capacity.

Table 1-1  MP-1 System Models

Figure 1-2 shows a typical MP 1200 series installation. The DPU is the large enclosure on the right; the DECstation is shown on the table. The storage expansion box and a CD-ROM reader can also be placed next to the DECstation. The MP 1100 series system looks similar, using a smaller DPU enclosure.
Figure 1-2  Typical MP-1 Installation
Front-End

The MasPar system front-end is a DECstation 5000 that runs ULTRIX (DEC’s implementation of the UNIX operating system) on a graphics workstation with windowing capability and standard I/O devices. This workstation provides the user with a keyboard and display, a network environment, a multiuser operating environment, a file system, a programming environment, and access to the DPU. The user process runs on the front-end UNIX operating system when a program runs on the DPU.

Figure 1-3 shows a diagram of the MasPar system with the front-end emphasized.

![Diagram](image)

Figure 1-3 MasPar System Diagram--DECstation 5000
Data Parallel Unit

The DPU is that part of the MasPar system that performs all the parallel processing. The DPU is composed of:

- The array control unit (ACU)
- A number of processor element array PCBs (PEs), from a minimum of one PCB (1,024 PEs) to a maximum of 16 PCBs (16,384 PEs)
- Processor element communication mechanisms

The array is arranged in a two-dimensional matrix. The DPU does not have its own operating system; it uses the front-end UNIX operating system.

The MasPar system includes two series of DPUs: the MP 1100-series and the MP 1200-series. The difference between these two series are the number of PE and I/O slots in the machine. The MP 1100-series has a maximum of 4 PE PCBs while the MP 12000-series has a maximum of 16 PE PCBs. More PEs means more data memory and higher performance. Also, the MP 1100-series has five available I/O slots while the MP 1200-series has 15 available I/O slots. The MP 1200 box is larger and requires more power than the MP 1100 box.

The number of processor elements is expandable in both MP 1100s and MP 1200s. There are 1,024 PEs on each PE PCB. The MP 1100 machines can be configured with 1, 2, or 4 PE PCBs; the MP 1200 machines can be configured with 1, 2, 4, 8, or 16 PE PCBs. If they have the same number of PEs, the MP 1100 machines are functionally the same as the MP 1200 machines.

For I/O, the DPU will have either a Parallel VME (PVME) PCB or an I/O Channel Controller (IOCTLR) PCB. If the IOCTLR is used, there may be one or more I/O RAM (IORAM) PCBs for additional IORAM.

Figure 1-4 shows a diagram of the MasPar system with the DPU that has a PVME installed emphasizing. Figure 1-5 shows the DPU that has an IOCTLR and one IORAM installed.
Figure 1-4  MasPar System Diagram--DPU with PVME

Figure 1-5  MasPar System Diagram--DPU with IOCTLR and an IORAM
DPU Card Cage

Figure 1-6 shows the card cage layout for both the MP 1100 and MP 1200. MasPar supplies the following printed circuit boards (PCBs) for the DPU (some of these PCBs are options):

- **Standard PCBs**
  - Array Control Unit (ACU)
  - Processor Element (PE) PCB
  - Router PCB

- **Optional PCBs**
  - Parallel VME (PVME) PCB, an I/O board
  - I/O Channel Controller (IOCTRL) PCB, an I/O board
  - I/O RAM (JORAM) PCB, an I/O board
  - Synchronous VME Interface PCB (SVME PCB), an I/O board that is part of the 3000-Series MasPar Disk Array Subsystem option. The SVME PCB sits in the MasPar Passive VME 6U Adapter, which adapts the VMEbus to the DPU backplane.
  - Asynchronous VME Interface PCB (AVME PCB), an I/O board that is part of the 4000-Series MasPar Disk Array Subsystem option. The AVME PCB sits in the MasPar VME 9U Adapter, which adapts the VMEbus to the DPU backplane.

- MVIB PCB
MP 1200 Card Cage

Figure 1-6  DPU Card Cages
Parallel VME (PVME) PCB

The DPU can have either a PVME PCB or an IOCTL PCB for I/O. The PVME PCB is a buffer between PEs and a VME device. The PVME is seen as a peripheral to the front-end workstation or to other I/O processors (IOPs) in the system. The VMEbus connects the PVME to these other processors. The PVME also has a RIO chip, connecting the PVME to the global router.

I/O RAM Buffer

The PVME has an I/O RAM buffer of 8 MBytes (expandable to 32 MBytes) of parity-checked memory.

The I/O RAM Buffer is available on the VMEbus as both a VMEbus master or VMEbus slave. Master operation, combined with a DMA controller, provides normal single-transfer bus operations and VME block-mode operations. The I/O RAM Buffer buffers transfers between the PE array and VMEbus.

The VMEbus port to the I/O RAM Buffer is capable of D32, D16, and D8 transfers. VME access to the I/O RAM Buffer includes a "byte swapping" feature to provide compatibility with little- and big-endian VME devices for byte, word, and longword object sizes.

VME Features

- I/O RAM Buffer and all registers fully memory-mapped into VMEbus address space.
- VMEbus interrupts for VME DMA transaction completion and error conditions.
- VME interface supports both A24 and A32 addressing modes as both master and slave. (A24 not supported with VAXstation 3520 as front end.)
- VME Block Mode access supported for all PVME addresses as slave.
- VME supports D32 and D16 transfers to all control registers as well as D32, D16, and D8 access to IORAM.
- Four modes of "byte swapping" supported by IORAM for compatibility with little- and big-endian VME devices.

Refer to the *MasPar Parallel VME Manual* for additional details of the PVME.
I/O Channel Controller (IOCTLR) PCB and I/O RAM (IORAM) PCB

The IOCTLR PCB is used instead of the PVME PCB for higher I/O throughput. The IOCTLR interfaces to:

- The MasPar Input/Output Channel (MPIOC), a high-speed bus used for transfers between I/O devices I/O RAM.
- The VMEbus with the same features as the PVME.
- The MasPar global router network that connects to the PEs. The IOCTLR and IORAM uses RIO chips to convert to/from the router’s serial data and the parallel data used by the MPIOC and VMEbus.

The IOCTLR has an I/O RAM Buffer of 8 MBytes of parity checked memory, expandable to 32 MBytes. For more I/O RAM, one or more I/O RAM (IORAM) PCBs are used. Each IORAM PCB has an I/O RAM Buffer of 32 MBytes of Error Code Correction (ECC)-protected memory, expandable to 128 MBytes.

See the *MasPar I/O Channel Controller and MasPar I/O RAM PCB Service Manual* for additional details of the IOCTLR, IORAM, and MPIOC.

MVIB PCB

The MVIB is a Digital Equipment Corporation (DEC)-supplied board that interfaces the DECstation 5000 to the VMEbus. The MVIB is a 9U VME board that connects to the DPU backplane by using the MasPar Passive VME 9U Adapter.

The MVIB acts as the DPU-end of the interface between the front-end and the DPU by interacting with the 3VIA board, which resides on the turbo-channel in the DECstation 5000.
The Array Control Unit (ACU)

The ACU is a register-based load/store processor with its own registers and data and instruction memory. It has a control processor, thirty-two 32-bit registers, 128 KBytes of data memory, 1 MByte of instruction memory RAM, and 4 GBytes of virtual instruction address space that is managed as 4096-byte pages.

The ACU does the following:
- Controls the PE array by sending data and instructions to each PE simultaneously
- Performs operations on singular data

It is possible to write a program that executes entirely in the DPU using MasPar Parallel Application Language (MPL). The ACU executes singular code and the PEs, under the control of the ACU, execute parallel code. Since the ACU is not as powerful as the DECstation 5000, it is not practical for most real applications to execute entirely in the DPU.

The ACU controls (sends instructions to) the PEs. Most singular code should be executed in the front-end, not the DPU. In some MasPar languages, the user must declare variables to be parallel variables or must make explicit calls to the DPU to get that instruction to execute in the DPU. Other MasPar languages have the compiler decide whether to use the DPU or the front-end; these decisions are largely transparent to the user.

PE PCB and Router PCB

Each PE PCB consists of 1,024 (1K) processor elements, 16 or 64 MBytes of memory (16 or 64 KBytes per PE), and three router chips, along with the necessary supporting logic.

Each Router PCB has one router chip with its supporting logic.

Refer to Page 1-13 for additional details of the PE PCB and the Router PCB.
PE Array

Each PE is a load/store arithmetic processor with dedicated register space and RAM. Each PE has a control processor, forty 32-bit private registers (the PREG), thirty-two 32-bit special purpose registers (the SREG) and either 16 or 64 KBytes of processor RAM (the PMEM). Figure 1-7 shows the block diagram of a PE.

![Block Diagram of a PE](image)

**Figure 1-7 PE Block Diagram**

The PE array is a two-dimensional matrix of PEs. The matrix has either has an equal number of columns and rows (a square matrix) or has twice as many columns as rows (a rectangle). The system’s matrix contains 1K, 2K, 4K, 8K, or 16K PEs.

A cluster is a non-overlapping square matrix of 16 PEs in the PE array. A 1K PE array is made up of 64 4x4 PE clusters, a 2K PE array has 128 clusters, and so forth. PE clusters are important in global router communications since there is only one global router connection to each cluster.

All parallel data is processed in the PE array. Parallel variables are declared by either the programmer or the compiler. The parallel variables are allocated on the PEs. Each PE receives the same instruction simultaneously from the ACU. PEs that are enabled by a data-dependent condition code then execute the instruction on these parallel variables. In this way, PEs are used to represent different data points in the same problem. For example, PEs can be used to represent the devices on a circuit, the pixels in an image, or the molecules in a fluid.

Figure 1-8 shows a diagram of the PE array.
Figure 1-8  The PE Array
PE Numbering

The physical layout of processors is a two-dimensional grid, with a row-column format. The origin of the grid is at the top left corner, with that processor being PE(0,0) using the notation PE(row,column). The number of columns is always either equal to the number of rows (a square array) or is twice the number of rows (a rectangular array).

The size of the row and column is determined by the system configuration. For example, a system with 16,384 PEs has 128 rows and 128 columns. There are five possible configurations:

- A Code 1 system has 1K PEs (one PE board) with 32 rows and 32 columns.
- A Code 2 system has 2K PEs (two PE boards) with 32 rows and 64 columns.
- A Code 3 system has 4K PEs (four PE boards) with 64 rows and 64 columns.
- A Code 4 system has 8K PEs (eight PE boards) with 64 rows and 128 columns.
- A Code 5 system has 16K PEs (16 PE boards) with 128 rows and 128 columns.

The five possible processor configurations are shown in Figure 1-9.

![Figure 1-9 System Configuration Codes](image-url)
PE Communications

There are two types of PE communications:

- communication between the ACU and the PEs
- communication between two PEs in the PE array

There are three mechanisms for implementing PE communications:

- ACU-PE Bus for communication between the ACU and the PEs
- X-Net for communication between two PEs in the PE array
- Global router for communication between two PEs in the PE array or between the PE array and the I/O section

ACU-PE Bus

Communications between the ACU and the PEs uses the ACU-PE bus. Most often, the communications originate with the ACU broadcasting instructions and data to the PEs.

When the PEs all simultaneously send a piece of data to some location in the ACU, the values of the parallel variables are logically reduced (global ORed) to one value in the singular ACU variable.

X-Net and Global Router Communications

Figure 1-10 shows the relationship between the X-Net and the Global Router communications. X-Net communications is discussed on Page 1-18 while the global router is discussed on Page 1-19.
Figure 1-10  X-Net and Global Router Communications
**X-Net**

The MP-1 uses *toroidal wrap* to model the layout of the PEs in the PE array. This means that all PEs have eight neighboring PEs, even PEs that are on an edge of the actual physical array.

X-Net communications are communications between any PE and any other PE that lies on a straight line from the original PE in one of the following directions: north, northeast, east, southeast, south, southwest, west, and northwest, even across PE PCBs. Figure 1-11 shows the eight direction lines that X-Net communications takes place.

X-Net communications can be incremented so that a PE can use X-Net communications to communicate with its $n$th neighbor to the north, for example.

![Diagram of X-Net](image)

*Figure 1-11  The Eight Directions on the X-Net*
Global Router

The global router provides connection between arbitrary sets of PEs or between PEs and an I/O device. The main difference between X-Net and global router communications is that the X-Net has a built-in direction of communications while the global router does not. In general, X-Net communications are faster than global router communications, but global router communications are more general purpose.

In MPL, the programmer chooses X-Net or global router communications explicitly by using different language constructs. In other MasPar languages, the communications mechanism is transparent to the user.

The global router uses 1024 bidirectional wires that transfer data in a serial half-duplex mode. Each cluster of 16 PEs shares a connection to a pair of router wires, one in and one out. Only 1 PE in the cluster can use the router at a time. All enabled clusters simultaneously send or receive messages. One PE in each cluster executes the same router instruction, and all participating PEs fetch or stores the same message bit at the same address on the same cycle.

The global router is three-staged, meaning that there are multiple redundant paths between stages. Router wires are first opened between clusters or between clusters and an I/O device, then data is transferred to or from the addressed PE.

The router chip performs switching. There is only one type of router chip. It’s placement in the global router network determines if it is a stage one, stage two, or stage three router chip. Each PE PCB has three router chips while each Router PCB has one router chip.

Stage one router chips acts as the source of the data for a data transfer. Each PE PCB has a stage one router chip that switches data from a specific PE in a cluster onto that cluster’s router wire.

Stage two router chips switches the connections so that there a complete path for the data transfer. In the MP 1200-series, there must be 16 stage two router chips on the backplane, one in each PE PCB slot. Each PE PCB and each Router PCB has a stage two router chip. Therefore, each PE PCB slot must have either a PE PCB or a Router PCB.

Stage three router chips acts as the destination of the data for a data transfer. Each PE PCB has a stage three router chip that switches data from the cluster’s router wire to a specific PE.

Figure 1-12 shows shows a Model 1202 2K DPU (two PE PCBs) with its global router network. The two PE PCB occupy slots 0 and 1; 14 Router PCBs are in slots 2 through 15.
Figure 1-12  Global Router Layout in a Model 1202 System
Disk Subsystem

The optional disk subsystem is the MasPar Disk Array (MPDA), which is a scaleable I/O device for bulk data storage. Architecturally, it is like a conventional BSD 4.2 UNIX file system and allows a variety of both performance and capacity options. It consists of a VME disk interface board that resides in the DPU and a parallel disk array. The VME disk interface board provides a data connection to the disk array.

The 3000-Series MPDA uses smaller disk drives in its array and has a slower data transfer rate than the 4000-Series. Refer to the *MasPar 3000-Series Parallel Disk Array Manual* and the *MasPar 4000-Series Parallel Disk Array Manual* for additional details of the MPDA.
Chapter 2

System Hardware Installation

The MP-1 physical system consists of a MasPar® DPU and a front-end DECstation 5000 with a storage expansion box. The following sections provide instructions for unpacking and installing the physical system. After these procedures are completed, you can power the system up, boot the system, and operate it. Figure 1-2 shows a typical installation. (Appendix A contains a brief installation checklist, without the details provided here.)

Refer to the MasPar 3000/4000-Series Parallel Disk Array Manual for MPDA unpacking and installation instructions.

Make sure your site meets the requirements listed in the MasPar Site Prep Guide.

**WARNING:** Turning on power before you are instructed to do so can cause injury or damage equipment.
Initial Inspection

Your shipment consists of several packages. The largest box on the pallet contains the MasPar DPU. Other boxes with DEC markings contain the DECstation 5000, video monitor, storage expansion box, etc. Smaller boxes contain accessories (tapes, cables, keyboard, and manuals) and any items packaged separately for special protection (for example, printed circuit boards (PCBs) for the DPU).

The following instructions for removing the DPU from its shipping container are also attached to the outside of the shipping container. Figures 2-1 through 2-3 show the unpacking procedure.

1. Inspect the container for obvious signs of damage, including dents, scrapes, or water damage, and note any sign of damage on the bill of lading.

2. Warning devices on the shipping container show if it was subjected to shock or tipped over. Another set of indicators is inside the container, on the DPU front panel. Inspect the outside indicators on receipt of the system. If the outside indicators are missing, unpack the system immediately, and inspect the inside set. Note any signs of shock or tipping on the bill of lading.
   - The Shockwatch indicator is a bright yellow and white square with a small, white vial in the center. If the vial is red instead of white, the container has been subjected to shock. Usually, this means that the box has been dropped.
   - The Tip-n-Tell is an orange square with an inverted "V" in the center. The bottom half of the "V" contains small blue particles; the top is light brown. If any of the blue particles are in the top of the "V," the shipping container was tipped.

3. If you find any sign of damage, or if any of the above indicators show that the package was shocked or tipped, note this on the bill of lading.
Unpacking the DPU

WARNING: The plastic bands are under tension and can snap or rebound when cut. Also, the ramp can fall when the bands and securing tape are cut.

1. Use a pair of scissors or a knife to cut the plastic bands around the box. Figure 2-1 shows the bands and the ramp.

2. While holding the ramp in place, cut the tape securing it to the side of the box.

3. Remove the ramp from the pallet and put it aside.

4. Remove the cardboard box:

   CAUTION: To avoid damage to the DPU, use a knife that does not penetrate more than 1 inch.
   a. Cut the tape along the seams on the top of the box.
   b. Release the two fasteners that are along the seam down the side of the shipping carton by squeezing or pinching the two flaps together until the fastener releases the cardboard. Figure 2-1 shows the fastener locations.
   c. Pull the box away from the DPU.

5. The DPU is now sitting on a pallet with boards around its base. Latches hold one of these boards in place, as shown in Figure 2-2.

   Open each latch by flipping its handle up so that it stands straight out, then turning it 1/4 turn counterclockwise.

6. Remove this board and place it aside, exposing the two U-bolts inserted in the pallet.

7. Lay the ramp against this side of the pallet so that the top of the ramp is flush with the surface of the pallet.

8. The two U-bolts in the pallet are used to fasten the ramp to the pallet. Lift out the U-bolts, and locate the matching holes at each side of the ramp and the pallet. Lower one U-bolt into each set of holes, straddling the gap between the ramp and the pallet. Figure 2-2 shows the U-bolt locations, and Figure 2-3 shows the U-bolts in the ramp.

   The U-bolts are not bolted into place; their own weight holds them in.

9. The ramp is now ready. Carefully roll the DPU off the pallet and down the ramp.

10. Gently push the DPU into position.

11. Set the levelers at each bottom corner of the enclosure. Spin each leveler down until it sits firmly against the floor, as shown in Figure 2-4.
While holding ramp, cut bands. **WARNING:** Bands can snap. Also, ramp may fall when bands are cut.

Place ramp aside. Cut tape on top of box. **CAUTION:** Knife should penetrate no more than 1 inch.

Release fasteners by pinching them towards their center. The box then comes apart at the seams.

Remove the box.

---

**Figure 2-1  Unopened Shipping Container**
Loosen latches and remove front 2x6 board. Open the latch key, so it stands away from the DPU, then turn it $\frac{1}{4}$ turn counterclockwise. This releases the board.

Figure 2-2  Removing the Front Board
Place the ramp so that the wide end is flush with the front of the pallet.

Remove the U-bolts from the pallet, and place them in the holes so that they connect the ramp to the pallet.

Roll the DPU down the ramp.

Figure 2-3  Ramp Ready to Use
Figure 2-4  Setting the Levelers
Unpacking the DEC DECstation 5000

Instructions for unpacking the DECstation are in the DECstation 5000 Hardware Installation Guide.

DECstation Installation

You should install the DECstation and verify its operation before beginning the DPU installation.

1. Install the DECstation, as described in detail in the DECstation 5000 Hardware Installation Guide. (You received a copy with the DECstation.) Installation involves setting up and cabling the system unit, video monitor, storage expansion box, optical disk reader, keyboard, and mouse.

2. Power-up the DECstation, to verify correct installation. This also initiates the power-up self-test described in the DECstation 5000 Hardware Installation Guide.

3. Power-down the DECstation before beginning the DPU installation procedures.
DPU Installation

**WARNING:** Turning on DPU power before you are instructed to do so can cause injury or damage equipment.

If you have not already done so, unpack the DPU. Unpacking instructions start on page 2-3.

Make sure the levelers at each corner of the DPU are set firmly against the floor.

Handling PCBs Correctly

**CAUTION:** To avoid damage, always handle a PCB correctly:

- Handle it gently by the edges. Never handle a PCB by the top or bottom surfaces.
- Store PCBs in a static-proof container.
- Wear antistatic equipment (for instance, a wrist or ankle strap) when handling the PCB.
- Use a static-free surface whenever working on a PCB.

Each PCB in the DPU card cage has ejector levers on the top and bottom of the board faceplate, shown in Figure 2-5. Always use the ejector levers when installing PCBs:

- To release the PCB, lift up the inner end of the top lever, and press down on the inner end of the bottom lever, moving the board partway out of its slot.
- To secure the PCB in the card cage slot, press down both the inner ends of the the top and bottom levers.

DPU PCBs

The DPU is shipped with the ACU, front-end VME interface, PE array boards, and router PCBs installed and held in place with three retaining bars.

**NOTE:**
Remove the retaining bars before removing the PCBs and reinstall the retaining bar after installing PCBs.

The sections below provide installation procedures for these boards.
ACU Jumper Settings

Before beginning the installation, verify the factory-installed ACU board jumper settings, as shown in Figure 2-6.

The ACU jumpers provide the following functions:

- Set the ACU VMEbus address.
- Set the ACU VMEbus interrupt request level.
- Set the ACU VMEbus interrupt acknowledge level.
- Select an independent scan path from the DECstation (for diagnostics only).
- Select the ACU as a VMEbus arbiter.
- Select either 16-bit or 32-bit words for the ACU instruction DMA.
- Select the ACU instruction DMA in either block or pipelined mode.
- Set the IMem size.
- Enable and disable the ACU board clock.
- Select the carry bit.
ACU Jumper Synopsis

The default values for the ACU jumper settings are as follows:

- ACU VMEbus address = 0xFCC000000
- ACU VMEbus interrupt level 1 selected
- ACU VMEbus arbiter selected
- OFF scan path selected
- Instruction DMA in pipelined mode
- 32-bit word
- Clock enabled
- IMem size = 1 MByte
Figure 2-6  ACU Jumpers

Note: Darkened jumpers are INSTALLED.
DPU Card Cage Access

The ACU, PE array, front-end VME interface, and router PCBs are accessed through the DPU rear door. Figure 2-7 shows the MP 1200 card cage access; the MP 1100 has similar access.

![Diagram of DPU Card Cage Access](image)

**Figure 2-7** Back of MP 1200 System and Card Cage

To open the DPU rear door, use the 5/16" hex wrench supplied with the system.
DPU Card Cage Slots

The DPU card cage is divided into two dedicated blocks: one for I/O boards and the other for PE array and router boards, as shown in Figure 2-8.

MP 1100 Card Cage

MP 1200 Card Cage

Figure 2-8  DPU Card Cage Slots

In the I/O section, the first slot on the left always holds the ACU board. The second I/O slot is configured to support an alternate VMEbus arbiter. Systems that have a DECstation 5000 as a front-end have the MVIB in this slot. The remaining I/O slots are for boards that connect to the MPIOC and/or the VMEbus.

The PE array boards must be in contiguous slots, starting with the first PE array board slot. (If the number of PE array boards changes, the system must be rejumpered, as described in Chapter 6.)

**CAUTION:** All PE array board slots that do NOT contain a PE array board must contain a router board. The MP-1 will not work if any PE array board slot is empty.

**CAUTION:** A PE array board cannot replace a router board nor can a router board replace a PE array board.
Installing the PCBs

Follow the steps to install the DPU PCBs.

**CAUTION:** Although you might have to push firmly to seat a board, avoid forcing PCBs when installing them.

1. Open the DPU rear door.
2. Remove the retaining bars.
3. Remove the ACU board.
4. Verify the jumper configuration on the ACU board, shown in Figure 2-6. The darkened blocks indicate installed jumpers.
5. Make sure the ACU crystal oscillator is firmly seated on the board.
6. Carefully slide the ACU board back into the slot.
7. With the ejector levers open (out), seat the board, and push the levers closed.
8. Secure the board in the card cage.
9. Replace the retaining bars
DPU Backplane Jumpers

The DPU backplane has jumpers installed for both the I/O and the PE array slots. Refer to Chapter 6 for descriptions of the PE array Xnet jumpers. (The Xnet jumpers are factory-set for your configuration as shipped.)

Referring to Figure 2-9 and the figure below, and going from right to left, install jumpers on all unused I/O slots, up to the last I/O board. Except for the ACU and alternate VME arbiter slots, do not install jumpers on occupied I/O slots.
MP 1200 Backplane

MP 1100 Backplane

Figure 2-9 DPU Backplane Jumpers
Connecting the DPU and DECstation Cables

This section provides procedures for connecting the DPU and DECstation cables. Figure 2-10 provides a simplified diagram of the DPU-DECstation cabling.

**CAUTION:** Inspect all cable connectors to make sure all the pins are straight. If any pin is bent, do not try to install it; replace the cable.

From the front, open the DPU front and inner front doors, and make sure the DPU backplane jumpers are installed correctly, as described above.

Connecting the Internal DPU Cables

After installing the DPU PCBs, follow the steps below to connect the internal DPU cables.

1. Connect the 100-pin AMP connector (labeled J3) of the 4-foot data cable to the 100-pin connector on the front-end VME interface PCB face plate.

2. Connect the data cable’s other 100-pin AMP connector (labeled JB) to the DPU interface PCB behind the rear cable panel:
   - MP 1200: Connector labeled P0A
   - MP 1100: Connector labeled P0B

3. Connect one end of the 2-foot internal telephone cable to the modem connection in the DPU.

4. Connect the other end of the internal telephone cable to the DPU interface PCB behind the rear cable panel.

![Diagram of DPU-DECstation Cables](image)

**Figure 2-10  DPU-DECstation Cables**
Connecting the External DPU Cables

Two cables connect the DPU and the DECstation. Figure 2-11 shows the DECstation connectors.

1. Connect the 100-pin AMP connector (labeled A1) of the 15-foot data cable to the 100-pin connector on the DPU lower rear cable panel.

2. Connect the 100-pin AMP connector (labeled P1) of the data cable to the 100-pin connector (port 1) on the back of the DECstation.

3. For North American systems only, connect the end of the 15-foot RS-232 cable to the 25-pin connector on the DPU lower rear cable panel.

4. For North American systems only, connect the other end of the RS-232 cable to the 25-pin connector (serial port 2) on the back of the DECstation.

![DECstation Rear Panel Connectors](image)

Figure 2-11 DECstation Rear Panel Connectors

System Power Requirements

The MP 1100 DPU requires either a *dedicated* 15A, 110V circuit or a *dedicated* 10A, 220V or 240V circuit, depending on destination country.

The MP 1200 DPU requires a *dedicated* 30A, 220V circuit. In the United States, use a NEMA L6-30A 220V connector.

The DECstation requires either a *dedicated* 15A, 110V circuit, or a *dedicated* 10A, 220V or 240V circuit, depending on destination country.

**CAUTION:** The DPU and the DECstation each require *dedicated* power circuits. They cannot share a power circuit with any other device, including each other.
Preparing the System Modem

NOTE: This section applies only to systems in North America.

An X11 telephone jack on the DPU cable panel connects a 2500-style (normal analog) telephone circuit to the modem inside the DPU. MasPar uses this modem to access the system when providing customer support.

Systems shipped inside the USA are equipped with an internal modem. The modem cables are already installed, going from the interface panel to the modem.

Systems shipped outside the USA are equipped with connections for external modems. Modems can be purchased locally and installed by MasPar personnel.

The modem requires software configuration before in order to work correctly. Instructions are provided in the *MasPar System Administration Manual.*
Connecting the DPU Power Cable

**CAUTION:** The DPU and the DECstation each require dedicated power circuits. They cannot share a power circuit with any other device, including each other.

- MP 1100 systems are equipped with a standard power cable and connector. Connect one end of the cable to the MP 1100 and the other to the power supply. The proper power supply is one of the following, depending on country of destination:
  
  * dedicated 15A, 110V circuit or
  * dedicated 10A, 220V or 240V circuit

- MP 1200 systems shipped in the United States must connect to a dedicated NEMA L6-30A connector.

  On MP 1200 systems shipped outside the United States, the ends of the power connector are left bare. Depending on local codes, you can choose to do one of the following:
  
  * Connect the MasPar power cable to a local adapter.
  * Wire the ends of the MasPar power cable directly to a terminal block.
  * Replace the MasPar power cable with a local power cable.

  Appendix B provides the instructions for wiring the power cable ends and for replacing the power cable.

**WARNING:** To avoid personal injury or damage to the equipment, read the instructions in Appendix B carefully. Have all connections inspected by a qualified electrician who is familiar with local codes.

After your system administration procedures are done, the system is ready for use. You should also become familiar with the DPU controls and indicators described in Chapter 3, and then read the *MasPar System Overview* manual.

Power Supply Levels

While not essential, measuring and adjusting the power supply levels can prevent problems during power-up. Refer to Chapter 3.

After your system administration procedures are done, the system is ready for use. You should also become familiar with the DPU controls and indicators described in Chapters 3 and 4. Then read the *MasPar System Overview* manual.
Chapter 3

Controls and Power

This chapter provides descriptions and functional definitions of the MP-1 controls. This chapter also discusses adjusting the DPU power supply and system powering up and down.

Figures 3-1 and 3-2 show the locations of the DPU controls. The DPU controls are listed on page 3-2 and are described in detail in the following sections.

Figure 3-1  Controls (Front of DPU)
**Controls Overview**

Detailed descriptions of the DPU controls are in the sections below.

- **VMEbus Reset**
  - Pushing this button resets the VMEbus *only if the keyswitch is set at DIAGNOSTIC.*

- **Modem**
  - At CONNECT, it enables the internal system modem (USA only).
  - At DISCONNECT, it disables the modem.

- **Disconnect/Connect**
  - Controls power to the DPU and enables the VMEbus reset button.
  - It is described in Chapter 1 and below.

- **Power Selector**
  - A three-position toggle switch described on Page 3-3.

- **30A/15A Circuit Breaker**
  - Described on Page 3-3.

- **1A Circuit Breaker**
  - Described on Page 3-3.
DPU Power System

The DPU power system is complex, and you should understand it thoroughly before changing any settings.

**CAUTION:** To avoid damage to the system, do not power-down the DPU before bringing the DECstation to "console" or "single-user" mode.

The power system includes LEDs indicating the status of the various power supplies and a power sequencer to ensure that the power supplies start in the correct order.

The DPU power system includes the following controls:

- The **keyswitch** on the DPU front inner door has three settings:
  - OFF turns power off.
  - ON powers-up the DPU.
  - DIAGNOSTIC is similar to ON, but it also enables the VMEbus RESET button.

- The **power selector** is on the power tray panel (inside the DPU enclosure at the rear), above the rectangular extension connecting the power tray to the DPU lower-rear panel. It is a toggle switch that must be pulled out slightly before it can be moved. It has three positions:
  - LOCAL (middle position): Powers-up the DPU
  - OVERRIDE (straight up): Powers-up the DPU, regardless of other conditions. The system will not power-up if the temperature sensor has tripped or is missing, or if the power supply is in an over-current situation.

  **WARNING:** FOR FACTORY USE ONLY. DO NOT USE THIS SETTING. IT OVERRIDES CRITICAL SAFETY SYSTEMS.

  REMOTE (lower position): (Not used). The DPU will NOT power-up.

- Two **circuit breakers** are on the outside rear of the DPU and are shown in Figure 3-2.
  - The **rocker switch** next to the power cable is a circuit breaker that is ON when it is UP. The circuit breaker is rated at 30A for MP 1200 systems and 15A for MP 1100 systems. The amber lamp above the rocker switch indicates there is power on both sides of the circuit breaker when it is lit.

  - The **pushbutton** controls the current to the circuits controlling the power sequencer. The amber lamp above the pushbutton indicates there is power on both sides of the breaker. The pushbutton trips and pops out when it detects loads of more than 1A. Push it in to reset it.
• A **heat sensor** shuts down the power supply if it detects excess heat. The power status LED changes to amber. After cooling, the power supply will not start again until the keyswitch is switched to OFF and then back to ON or DIAGNOSTIC. An LED on the power supply also indicates excess temperature. (See page 4-7.)

• When the DPU is powered-on, the **power-up sequencer** turns on the –5V power supply and verifies its correct operation before turning on the +5V power supply. During this period (approximately 3 seconds), the power status LED is amber. When the sequencer turns on the +5V power supply, the power status LED turns green.

If the power status LED remains amber, the power-up sequencer did not complete its sequence successfully. If this happens, there may be a problem with the power tray. Troubleshooting procedures for the power tray is in Chapter 4.
Normal Operation

To power-up the DPU, follow the steps below:

1. Make sure the DECstation is powered-down.

2. Make sure the DPU circuit breakers are ON (the rocker switch is UP, and the pushbutton is IN).

3. Set the power selector to LOCAL.

4. Turn the keyswitch to ON.

   The power status LED is amber for approximately 3 seconds, then turns green. You will also hear the fan start.

5. Power-on the DECstation.

Table 3-1 lists all possible Power Selector/Keyswitch combinations.

<table>
<thead>
<tr>
<th>Power Selector</th>
<th>Keyswitch</th>
<th>DPU Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCAL</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>LOCAL</td>
<td>ON/DIAGNOSTIC</td>
<td>ON</td>
</tr>
<tr>
<td>OVERRIDE</td>
<td>N/A</td>
<td>ON (Factory Use Only)</td>
</tr>
<tr>
<td>REMOTE</td>
<td>N/A</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Table 3-1  DPU Power Settings
Adjusting DPU Power Supply

This section describes how to measure and adjust the power supply voltage levels.

The DPU cabinet has four power supplies: +5V, −5.2V, +12V, and −12V. Measure these voltages on the DPU backplane, as shown in Figure 3-3, and adjust the levels on the points shown in Figures 3-5 and 3-4.

**WARNING:** Do not short the power supply leads to the ground or to other power supply leads. When working around the power supply, do not wear loose clothing or jewelry, especially watches or rings. Failure to observe these precautions can cause personal injury and damage the equipment.

For all voltage measurements and adjustments:

- Use a digital voltmeter to measure the supplies. Adjust them only if they measure beyond the tolerances specified in the following paragraphs.

- Measure the supplies at the points shown in Figure 3-3.

- Use an insulated screwdriver to make any adjustment.
Figure 3-3  DPU Backplane Voltage Test Points
+5V Supply
- Adjust to exactly 5.0 V.
- Adjust only if it is outside the range of +4.75 to +5.25 V.

−5.2V Supply
- Adjust to exactly −5.2 V.
- Adjust only if it is outside the range of −5.45 to −4.95 V.

−12V Supply
- Adjust to exactly −12 V.
- Adjust only if it is outside the range of −12.25 to −11.75 V.

+12V Supply
- Adjust to exactly +12 V.
- Adjust only if it is outside the range of +12.15 to +11.85 V.

The following figures show the power supply adjustment points and wiring for the MP 1100 and the MP 1200 DPUs.

NOTE: The MP 1100 can use either of two types of power supplies, as shown in Figure 3-4.
MP 1100 HC Power Supply

MP 1100 Powertec Power Supply

Caution: Overtightening the 12V wires can break the posts on the power supply.

Figure 3-4 MP 1100 Power Supply Voltage Adjustments
Caution: Overtightening the 12V wires can break the posts on the power supply.

Figure 3-5 MP 1200 Power Supply Wiring and Voltage Adjustments
Powering the System Up and Down

The DECstation power switch is on the back of the unit, as shown in Figure 3-6. It is a rocker switch, labeled 1 and 0. The DECstation is ON when the 1 is in and the 0 is out.

![DECstation Rear Panel Connectors](image)

100-pin Data Cable
25-pin RS232 Cable
ON/OFF Switch

Figure 3-6 DECstation Rear Panel Connectors

The DPU has two switches controlling power: the keyswitch and the power selector. Figures 3-1 and 3-2 show their locations and page 3-2 provides details on these switches.

Power-Up Sequence

**CAUTION:** To avoid damage to the system, always power-up or power-down in the correct sequence.

1. Power-up the DPU.
2. Power-up the DECstation.
3. Boot the DECstation.

When the system boots, it is ready to operate.

**NOTE:** If you boot the DECstation before you power up the DPU, you cannot access the DPU. Any time you reconnect or repower the DPU, you must reboot the DECstation.
Power-Down Sequence

1. Halt the DECstation, using either /etc/halt or /etc/shutdown.

2. Power-down the DPU, setting the keyswitch to OFF or setting 30A breaker to OFF (down). Always set the 30A breaker OFF when working on the power supply.

3. Power-down the DECstation, as described in the DECstation 5000 Hardware Installation Guide.
There are three types of troubleshooting tools for the MasPar® MP-1 DPU:

- LEDs
- Error Logs
- Diagnostic Software
DPU LEDs

Three areas of the DPU have status LEDs. They are the:

- LED indicators on the front of the DPU
- LEDs on the PCBs
- LEDs on the power supply

DPU Front Panel LEDs

Figure 4-1 shows the DPU front panel and Table 4-1 lists the LEDs.

![Diagram of DPU Front Panel LEDs]

Figure 4-1  DPU Front Panel LEDs
<table>
<thead>
<tr>
<th>Label</th>
<th>Green</th>
<th>Amber</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power</td>
<td>Power, fan, and temperature are OK</td>
<td>System powering-up*</td>
<td>No AC power present</td>
</tr>
<tr>
<td>19</td>
<td>Macrocode is loaded</td>
<td>Macrocode is not running</td>
<td>(Not used)</td>
</tr>
<tr>
<td>18</td>
<td>ACU is waiting for TOBEQ (to back end queue)</td>
<td>(Not used)</td>
<td>ACU is not waiting for TOBEQ</td>
</tr>
<tr>
<td>17</td>
<td>ACU is waiting for FRBEQ (from back end queue)</td>
<td>Page fault (overrides green)</td>
<td>ACU is not waiting for FRBEQ</td>
</tr>
<tr>
<td>16</td>
<td>PMEM is using PE</td>
<td>PMEM is not using PE</td>
<td>(Not used)</td>
</tr>
<tr>
<td>15</td>
<td>Router is active</td>
<td>(Not used)</td>
<td>No router activity</td>
</tr>
<tr>
<td>14</td>
<td>I/O is taking place between PE(s) and I/O device(s)</td>
<td>Machine is temporarily stalled due to register interlock</td>
<td>One or more PEs are selected.</td>
</tr>
<tr>
<td>13</td>
<td>VME AS is active</td>
<td>(Not used)</td>
<td>VME AS is not active</td>
</tr>
<tr>
<td>12</td>
<td>VME DTACK is active</td>
<td>(Not used)</td>
<td>VME DTACK is not active</td>
</tr>
<tr>
<td>11</td>
<td>(Not used)</td>
<td>VME parity error</td>
<td>VME parity OK</td>
</tr>
<tr>
<td>10</td>
<td>(Not used)</td>
<td>(Default)</td>
<td>(Not used)</td>
</tr>
<tr>
<td>9-6</td>
<td>(Not used)</td>
<td>(Not used)</td>
<td>(Not used)</td>
</tr>
<tr>
<td>5</td>
<td>Background Diagnostic Running</td>
<td>Background Diagnostic Failure Check the dpujobmgr error log, referring to Page 4-8</td>
<td>idle</td>
</tr>
<tr>
<td>4-0</td>
<td>SCROLLING Dpumanager is running</td>
<td>(Not used)</td>
<td>Dpumanager not running**</td>
</tr>
</tbody>
</table>

*During normal system power-up, the power status LED is amber for a few seconds, then changes to green. If it remains amber, the power sequencer did not complete power-up. Check power supply LEDs, referring to Page 4-7.

**One of the five LEDs will be ON during the running of Background Diagnostics, which executes about once every ten seconds when the DPU is idle.

Table 4-1 Front Panel LEDs
Printed Circuit Board LEDs

Figure 4-2 shows the LEDs on the PCBs.

**MP 1200 Card Cage**

![Diagram of MP 1200 Card Cage]

**MP 1100 Card Cage**

![Diagram of MP 1100 Card Cage]

- **ACU LEDs**
  - Power ON
  - Bus Grant
  - Microcode Interrupt
  - Any_Reg
  - Data Strobe
  - Address Strobe
  - DTACK
  - IBUSY
  - BUSERR
  - MMSEL
  - Mempty
  - IFUVAL

- **PE Board LEDs**
  - Power ON
  - Parity Error
  - Selected for diagnostics
  - GOR enabled
  - GOR bit 0
  - GOR bit 1
  - GOR bit 2
  - GOR bit 3

- **Router Board LEDs**
  - Power ON
  - Parity Error
  - Selected for diagnostics
  - GOR enabled
  - (GOR = Global OR)

Figure 4-2  Printed Circuit Board LEDs
Table 4-2 lists the ACU LEDs.

<table>
<thead>
<tr>
<th>LED</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power ON</td>
<td>ON when the ACU board is powered</td>
</tr>
<tr>
<td>Bus Grant</td>
<td>ON when the ACU has VMEbus grant</td>
</tr>
<tr>
<td>Microcode Interrupt</td>
<td>ON when a microcode interrupt is in progress</td>
</tr>
<tr>
<td>Any_Reg</td>
<td>ON during a valid ACU access over the VMEbus</td>
</tr>
<tr>
<td>DS</td>
<td>VME data strobe</td>
</tr>
<tr>
<td>AS</td>
<td>VME address strobe</td>
</tr>
<tr>
<td>DTACK</td>
<td>ACU is generating DTACK</td>
</tr>
<tr>
<td>IBUSY</td>
<td>ACU is master of the bus</td>
</tr>
<tr>
<td>BUSERR</td>
<td>When ON, a VMEbus transaction did not complete or completed with an error. (An addressed VMEbus device did not respond within the VMEbus timeout limit (~ 60 µsec) or returned an error signal in response to a VMEbus access.)</td>
</tr>
<tr>
<td>MMSEL</td>
<td>When ON, ACU issuing current microcode from the M machine. (The current operation is a memory access, not a PE calculation.)</td>
</tr>
<tr>
<td>Mempty</td>
<td>When ON, no pending memory operations in the M machine</td>
</tr>
<tr>
<td>IFUVAL</td>
<td>When ON, instruction fetch unit (IFU) directed to valid address. Reasons for an invalid IFU access:</td>
</tr>
<tr>
<td></td>
<td>1. Attempting to execute code while a refresh cycle is in progress.</td>
</tr>
<tr>
<td></td>
<td>2. A page fault.</td>
</tr>
<tr>
<td></td>
<td>3. Attempting to execute code while the VMEbus is talking to instruction memory.</td>
</tr>
</tbody>
</table>

Table 4-2  ACU LEDs
Each PE array board has eight LEDs arranged into two 4-LED banks. The upper bank provides status information about the board, and the lower bank displays the results of the global OR (GOR) for that board. Router boards have only the status LEDs.

Table 4-3 lists the LEDs from the top.

<table>
<thead>
<tr>
<th>LED</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power ON</td>
<td>ON to indicate that the board is powered.</td>
</tr>
<tr>
<td>Parity Error</td>
<td>ON when a parity error occurs, and OFF when the error-handling routine is completed or cleared by a reset.</td>
</tr>
<tr>
<td>Selected for Diagnostics</td>
<td>ON when the board is selected for diagnostics.</td>
</tr>
<tr>
<td>GOR Enabled</td>
<td>ON when the board GOR utility is selected.</td>
</tr>
<tr>
<td>PE array boards only:</td>
<td></td>
</tr>
<tr>
<td>GOR bit 0</td>
<td>ON when GOR bit 0 is active.</td>
</tr>
<tr>
<td>GOR bit 1</td>
<td>ON when GOR bit 1 is active.</td>
</tr>
<tr>
<td>GOR bit 2</td>
<td>ON when GOR bit 2 is active.</td>
</tr>
<tr>
<td>GOR bit 3</td>
<td>ON when GOR bit 3 is active.</td>
</tr>
</tbody>
</table>

Table 4-3  PE and Router LEDs

If all the PCBs have their Power ON LEDs OFF, check power supply LEDs, referring to Page 4-7.

If only one PCB has its Power ON LED OFF, replace that board.

Parity Errors are logged in the error logs. See Page 4-8.
Power Supply LEDs

Figure 4-3 shows the eight LEDs on the DPU power tray rear panel. These LEDs are inside the DPU and provide information about the power supply. They all show the status of some part of the power supply and are green during normal operation. If a failure is detected, the corresponding LED(s) change(s) to red, and the power supply shuts down. Red LEDs latch ON, indicating the problem area initiating the power shut-down.

![Diagram of Power Supply LEDs]

Power tray (inside DPU cabinet)  
Other components not shown.

Power selector

Figure 4-3  Power Supply LEDs

From the top, the power supply LEDs are as follows:

- Chassis Ground (See note.)
- Temperature
- Main power Vcc +5V
- Main power Vee −5V
- Main power +12V
- Main power −12V
- Auxiliary power Vcc +5V
- Auxiliary power Vee −5V
- Auxiliary power +12V
- Auxiliary power −12V

NOTE:
The chassis ground circuit detects differences in potential between chassis ground and logic ground and shuts down the power supply when the potential exceeds a preset threshold.
Interpreting Log File Messages

Two logs give useful information about the status of the DPU during normal operation:

/usr/adm/dpujobmgr.log

/etc/uerf

In addition, a third log (. /LOG), records the output of most diagnostic programs. Diagnostics should not be run when other users are on the system because diagnostic programs need all the PMEM. See Page 4-14 for a discussion on this log and the diagnostic programs.

/usr/adm/dpujobmgr.log

This text file is a log of all the dpumanager daemon activity (the dpumanager daemon must be running). This log reports background diagnostic errors, register status when errors are reported, and some ACU kernel information. This log is especially helpful when you are trying to determine what might have caused a problem when a program aborts.

This example shows that there was a control store PE parity error (FLTCOD=2).

(dpu0) Tue Apr 30 07:04:11 1991 (81) DPU fault: SWOPT=5; HWOPT=5; FLTCOD=0x2; PMSTAT=0x0; PMEMECC=0

The mnemonics for this example are described below:

SWOPT The DPU size that the program running is using when the error occurred. The range is from 1 to 5. It is possible that the program uses fewer PEs than the DPU physically contains. The DPU PE numbering scheme is explained below.

HWOPT The physical DPU size. A program is allowed to use fewer PEs than the DPU physically contains. The codes are the same as SWOPT, above.

FLTCOD Fault Code Word. Its value provides a probable cause of an error. Table 4-4 lists the fault codes.

PMSTAT PE Memory Status. Its value is explained in Table 4-4.

PMEMECC The number of PE Memory ECC retries during a transaction. The limit is five. A sixth error aborts the program.
**PE Numbering**

The size of the row and column is determined by the system configuration. For example, a system with 16,384 PEs has 128 rows and 128 columns. There are five possible configurations:

- A HWOPT Code 1 system has 1K PEs (one PE board) with 32 rows and 32 columns.
- A HWOPT Code 2 system has 2K PEs (two PE boards) with 32 rows and 64 columns.
- A HWOPT Code 3 system has 4K PEs (four PE boards) with 64 rows and 64 columns.
- A HWOPT Code 4 system has 8K PEs (eight PE boards) with 64 rows and 128 columns.
- A HWOPT Code 5 system has 16K PEs (16 PE boards) with 128 rows and 128 columns.

The five possible processor configurations are shown in Figure 4-4.

![Diagram of PE Numbering and Configuration Codes](image-url)

**Figure 4-4  System Configuration Codes**
<table>
<thead>
<tr>
<th>Value</th>
<th>Error</th>
<th>Meaning</th>
<th>Probable Cause*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td>CLTSTOPARITY</td>
<td>Control store parity error</td>
<td>ACU</td>
</tr>
<tr>
<td>0x2</td>
<td>CTLPEPARITY</td>
<td>Control store PE parity error</td>
<td>PE, ACU</td>
</tr>
<tr>
<td>0x4</td>
<td>CTLIOPARITY</td>
<td>Control store I/O parity error</td>
<td>ACU, I/O device</td>
</tr>
<tr>
<td>0x10</td>
<td>CMEMLIMIT</td>
<td>CMem limit violation</td>
<td>Software, ACU</td>
</tr>
<tr>
<td>0x20</td>
<td>PREGLIMIT</td>
<td>PReg limit violation</td>
<td>Software, PE, ACU</td>
</tr>
<tr>
<td>0x40</td>
<td>PMEMLIMIT</td>
<td>PMEM limit violation</td>
<td>Software, PE, ACU</td>
</tr>
<tr>
<td>0x100</td>
<td>IMEMPARITY</td>
<td>Instruction memory parity error</td>
<td>ACU</td>
</tr>
<tr>
<td>0x200</td>
<td>RTRPARITY</td>
<td>Router parity error</td>
<td>PE, Router, ACU, RIO device</td>
</tr>
<tr>
<td>0x400</td>
<td>XNETPARITY</td>
<td>Xnet parity error</td>
<td>PE, ACU</td>
</tr>
<tr>
<td>0x800</td>
<td>RTRTNOR</td>
<td>Router transmit no receive</td>
<td>PE, Router, ACU</td>
</tr>
<tr>
<td>0x1000</td>
<td>PREGALIGN</td>
<td>PReg alignment</td>
<td>Software</td>
</tr>
<tr>
<td>0x2000</td>
<td>IMEMALIGN</td>
<td>IMem alignment</td>
<td>Software</td>
</tr>
<tr>
<td>0x4000</td>
<td>CMEMALIGN</td>
<td>CMem alignment</td>
<td>Software</td>
</tr>
<tr>
<td>0x10000</td>
<td>ILLOP</td>
<td>Illegal operation</td>
<td>Software</td>
</tr>
<tr>
<td>0x80000</td>
<td>BUSERR</td>
<td>VMEbus error</td>
<td>I/O device**</td>
</tr>
<tr>
<td>0x100000</td>
<td>MACHFAULT</td>
<td>Machine fault</td>
<td>ACU, Software</td>
</tr>
<tr>
<td>0x200000</td>
<td>IOERR</td>
<td>I/O instruction error</td>
<td>I/O device, ACU***</td>
</tr>
<tr>
<td>0x400000</td>
<td>RPROTO</td>
<td>Router protocol error</td>
<td>Software, MVIB, PE, Router, ACU</td>
</tr>
<tr>
<td>0x800000</td>
<td>IILLEPADDR</td>
<td>Illegal PE address</td>
<td>Software</td>
</tr>
<tr>
<td>0x1000000</td>
<td>DIV0</td>
<td>Integer divide by zero</td>
<td>Software</td>
</tr>
<tr>
<td>0x8000000</td>
<td>FINEXACT</td>
<td>Floating-point inexact result</td>
<td>Software</td>
</tr>
<tr>
<td>0x10000000</td>
<td>FINVOP</td>
<td>Floating-point invalid operand</td>
<td>Software</td>
</tr>
<tr>
<td>0x200000000</td>
<td>FDIV0</td>
<td>Floating-point divide by zero</td>
<td>Software</td>
</tr>
<tr>
<td>0x4000000000</td>
<td>FUNFLO</td>
<td>Floating-point underflow</td>
<td>Software</td>
</tr>
<tr>
<td>0x80000000000</td>
<td>FOVFLO</td>
<td>Floating-point overflow</td>
<td>Software</td>
</tr>
</tbody>
</table>

*ordered by greatest probability to least probability

**other than ACU or MVIB

***if FLTCOD=200200 (200000+200) then PE, router

Table 4-4  FLTCOD Values
### Table 4-5 PMSTAT Values

When `dpumanager` is running and the DPU is idle, `dpumanager` executes the DPU background diagnostics about once every ten seconds. Errors found by the background diagnostics are logged in the `/usr/adm/dpujobmgr.log`.

The background diagnostics generate two error codes:
- Code 1 means a failure in the X-Net test
- Code 2 means a failure in the Router test

This example shows an execution of a "mpshutdown command" and the execution of a "dpumanager" command.

```
(dpu0) Mon May 13 10:39:44 1991 Termination signal received; shutting down
(dpu0) Mon May 13 10:52:48 1991 Starting up; Version 2.1.74
(dpu0) Mon May 13 10:52:50 1991 loading microcode file:
"/usr/mspar/mp12ucode.wo"
(dpu0) Mon May 13 10:53:02 1991 loading ACU kernel file:
"/usr/mspar/etc/acuk"
```

This example shows a hang-up with the ACU kernel. The kernel timed out and automatically restarted.

```
(dpu0) Tue Apr 30 15:43:00 1991 ACU kernel timeout (command 15)
ECSR=0x4002, QCSR=0x0, PTACCESS=0, CPC=0xffff00a8
(dpu0) Tue Apr 30 15:43:00 1991 loading ACU kernel file:
"/usr/mspar/etc/acuk"
```

This example shows the log after a correctable PMEM error occurred. If PMEM=6 or more, a hard error is recorded, and the program aborts.

```
(dpu0) Mon May 13 05:37:49 1991 (82) 1 PE had errors
(dpu0) Mon May 13 05:37:49 1991 (82) PE fault: PE number=0xa5f (board 4, cluster 5,7, PE-in-cluster 0,3); error bits=0x8 (PMEM/SOFT)
(dpu0) Mon May 13 06:46:01 1991 (84) PMEM parity errors detected;
PMEM=1 (in rollpmem)
```

"in rollpmem" indicates that the memory error occurred during job swapping.
/etc/uerf

This is the ultrix error formatter. All front-end, ACU, and I/O errors are reported in this log. This is very helpful when the dpumanager is not running. Refer to man pages for information on how to use all of the features of uerf.

To look at the log, enter the following commands:

```bash
# cd /etc
# uerf -R | more
```

In the log examples below, Example 1 is a message from the DPU; FLTCOD 400 indicates an Xnet parity error. Example 2 is an example of an ULTRIX error message; it has nothing to do with the DPU, but is shown here as a contrast to the DPU message. Example 3 is a keyboard error.

**Example 1:**

```plaintext
----- EVENT INFORMATION -----  OPERATIONAL EVENT
EVENT CLASS                  ASCII MSG
OS EVENT TYPE                250.
SEQUENCE NUMBER              160.
OPERATING SYSTEM             ULTRIX 32
OCCURRED/LOGGED ON           Wed May 8 14:09:49 1991 PDT
OCCURRED ON SYSTEM           showl
SYSTEM ID                    x0A000005
SYSTYPE REG.                  x03010002
FIRMWARE REV = 5.
PROCESSOR TYPE               x00000008
PROCESSOR COUNT              pc=0xffff0700, fltcod=0x400
PROCESSOR WHO LOGGED         KA60
MESSAGE                      tcp_input: so->ref: ld
```

**Example 2:**

```plaintext
----- EVENT INFORMATION -----  OPERATIONAL EVENT
EVENT CLASS                  ASCII MSG
OS EVENT TYPE                250.
SEQUENCE NUMBER              159.
OPERATING SYSTEM             ULTRIX 32
OCCURRED/LOGGED ON           Tue May 7 15:27:33 1991 PDT
OCCURRED ON SYSTEM           showl
SYSTEM ID                    x0A000005
SYSTYPE REG.                  x03010002
FIRMWARE REV = 5.
PROCESSOR TYPE               x00000008
PROCESSOR COUNT              KA60
PROCESSOR WHO LOGGED         tcp_input: so->ref: ld
MESSAGE                      tcp_input: so->ref: ld
```
Example 3:

```
uerf -R -o terse | more

version 4.2-011 (15)

MESSAGE
ws: Keyboard error, code = b6
MESSAGE
MPFS: 20 buffers of 262144 bytes each
SYSTEM STARTUP
Fri Sep 13 11:18:49 1991
MESSAGE
ULTRIX V4.2 (Rev. MP-2.1A) System #2: Mon Aug 26
     21:41:35 PDT 1991
1. real mem = 25165824
   avail mem = 19611648
   using 614 buffers containing 2514944 bytes of memory
   KN02 processor - system rev 46
   cpu0 ( version 3.0, implementation 2)
   fp0 ( version 3.0, implementation 3)
   asc0 at ibus5
2. rz0 at asc0 slave 0 (RZ56)
3. rz4 at asc0 slave 4 (RRD40)
   ln0 at ibus6
   ln0: DEC LANCE Ethernet Interface, hardware address:
   08:00:2b:1c:83:94
   dc0 at ibus7
   fb0 at ibus0
5. vba0 at slot 2 (3VIA/MVIB)
6. ms0 at vba0 csr 0xfc800000 vmea32d32 vec 0xbe priority
   1
7. da0 at ms0 slave 0
   (D515-78 - 8 data drives)
8. acu0 at vba0 csr 0xfcc00000 vmea32d32 vec 0xd0 priority
    1
   iqprobe at be889000
9. iq0 at vba0 csr 0xdc000000 vmea32d32 vec 0xc0 priority
   1
```

1. real mem = Physical memory installed in the workstation.

2. rz0 = Internal disk drive

3. rz4 = CD reader

4. Ethernet Address

5. vba0 = DS5000-to-DPU interface

6. ms0 = Disk array controller

7. da0 = Bank 1

8. acu0 = DPU's ACU

9. iq0 = PVME or IOCTLR
Using Diagnostic Software

MasPar provides diagnostics and related utility programs to test the ACU board, the PE array board, the PE array/router connections, the router PCB, EEPROMs, and memory. For information on MasPar Parallel Disk Array diagnostics, refer to the *MasPar Parallel Disk Array Manual*.

**CAUTIONS:**

1. Before running MasPar diagnostics, make sure the front-end system is running correctly.

2. Make sure no other users are using the system. MasPar diagnostics uses all the PMEM if dpumanager is running.

3. While it is very unusual, a few diagnostic programs might cause the system to crash.

4. To abort a diagnostic program while it is running, do not kill the process. Suspend it by typing Ctrl-C (see below for details); this allows the diagnostic to restore the system to a usable state.

The diagnostic programs run under the ULTRIX operating system, and reside in the directory $MP\_PATH/field/bin. The diagnostics require maspar or root privileges or their equivalent.

**NOTE:** Type `setenv DIAG\_PATH $MP\_PATH` before running diagnostics.

To run a diagnostic test type the name of the test followed by a space and any desired option, then press Return. The default is a "verbose" mode that displays all related messages. The options are as follows:

- `-q` *quick* mode; some lengthy tests are shortened.
- `-t` *terse* mode; only the diagnostic summary is displayed.
- `-v` *verbose* mode; results of individual test and the diagnostic summary are displayed.

- `-M` *Monitor Mode*
To suspend a diagnostic test, type Ctrl-C (^C). This suspends the current test and displays a menu similar to the following example:

**TEST SUSPENDED AT USER’S REQUEST**

1. Exit to the calling shell
2. Continue with the test

Select one:

Type 1 or 2 to make the selection. The selections have the following effects:

1. Exit to the calling shell: In most cases, the calling shell is the UNIX system command processor. In some cases, the calling shell is a higher-level diagnostic that calls the next lower-level diagnostic. In such cases, continue to use Ctrl-C until the UNIX system command processor is reached.

2. Continue with the test: The diagnostic continues exactly what it was doing when Ctrl-C was pressed.

**Diagnostic Messages**

Following are standard diagnostic file messages and brief explanations.

<table>
<thead>
<tr>
<th>CAUTION</th>
<th>An abnormal result, but not an actual error. This could indicate trouble elsewhere; this is often the case when there are unused bits not at 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED</td>
<td>The reporting test received one or more ERRORs.</td>
</tr>
<tr>
<td>ABORTED</td>
<td>The test terminated abnormally before completion.</td>
</tr>
<tr>
<td>WARNING</td>
<td>The program encountered a serious situation associated with an area not being directly tested. This situation will not necessarily cause the current test to fail, but it can provide a clue to other problems. This is a more serious message than CAUTION.</td>
</tr>
<tr>
<td>PASSED</td>
<td>The selected test completed successfully.</td>
</tr>
</tbody>
</table>
MP-1 diagnostics consist of two sets of programs: test suites and individual tests. MasPar provides two tests suites, acu_diag and pe_diag. First run acu_diag, then pe_diag. Tables 4-6 through 4-7 describe the MP-1 diagnostic tests.

acu_diag

acu_diag tests the ACU board. It runs the following tests in the order shown:
1. acu_reg1
2. wcs memory test*
3. acu_micro
4. cmem memory test*
5. acu_reg2
6. imem memory test*
7. aux memory test*
8. map memory test*
9. acu_pgtbl
10. acu_macro
11. acu_bound
12. acu_pptest
13. acu_clim
14. pe_block

*Cannot be run outside of acu_diag.
pe_diag

pe_diag tests processor array boards and the backplane. It runs the following tests in the order shown:

1. pe_scan
2. pe_mem
3. pe_macro →
4. pe_arith
5. pe_func
6. pe_block
7. pe_xnet
8. pe_rtbp
9. pe_rtdiag
10. pe_rtr →

pe_misc
<table>
<thead>
<tr>
<th>Tests</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acu_bound</td>
<td>ACU Boundary Test—tests the ability of macro code to operate across page and row boundaries.</td>
</tr>
<tr>
<td>acu_clim</td>
<td>Tests the action of the ACU when certain CMEM limits and alignments are violated.</td>
</tr>
<tr>
<td>acu_diag</td>
<td>This test suite checks the ACU board. The tests in the suite are listed above.</td>
</tr>
<tr>
<td>acu_int</td>
<td>First of two tests of interrupt operation.</td>
</tr>
<tr>
<td>acu_int2</td>
<td>Second of two tests of interrupt operation.</td>
</tr>
<tr>
<td>acu_macro</td>
<td>Tests the ability of macro code to execute basic instructions.</td>
</tr>
<tr>
<td>acu_micro</td>
<td>Tests data paths, registers, and components on the ACU board.</td>
</tr>
<tr>
<td>acu_pgtbl</td>
<td>Tests the page table translation and comparison mechanism.</td>
</tr>
<tr>
<td>acu_pptest</td>
<td>Verifies the 32 general-purpose registers are functional; uses peeks and pokes.</td>
</tr>
<tr>
<td>acu_prof</td>
<td>Tests the three profile counters and the profile configuration register.</td>
</tr>
<tr>
<td>acu_reg1</td>
<td>Tests the ACU front-end registers.</td>
</tr>
<tr>
<td>acu_reg2</td>
<td>Tests the ACU front-end firmware-emulated registers.</td>
</tr>
<tr>
<td>acu_sup</td>
<td>Tests the hardware necessary to support user/supervisor mode firmware, CMEM address map, and software interrupt generation.</td>
</tr>
<tr>
<td>pe_arith</td>
<td>Tests PE arithmetic functions.</td>
</tr>
<tr>
<td>pe_block</td>
<td>Tests ability to make block transfers between the front-end memory and PMEM.</td>
</tr>
<tr>
<td>pe_diag</td>
<td>This test suite checks the PE array boards and the backplane. The tests in the suite are listed above.</td>
</tr>
<tr>
<td>pe_func</td>
<td>Tests various PE functions.</td>
</tr>
<tr>
<td>pe_macro</td>
<td>Tests the instruction set.</td>
</tr>
<tr>
<td>pe_memdiag</td>
<td>Tests PMEM and reports the exact location of any errors found.</td>
</tr>
<tr>
<td>pe_micro</td>
<td>Uses special microcode to test miscellaneous PE functions.</td>
</tr>
</tbody>
</table>

**Table 4-6  Diagnostic Tests, Sheet 1 of 2 Sheets**
<table>
<thead>
<tr>
<th>Tests</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pe_rtdiag</td>
<td>Tests the transmitting and receiving router functions.</td>
</tr>
<tr>
<td>pe_rtr</td>
<td>PE router wire test.</td>
</tr>
<tr>
<td>pe_rtrcnt</td>
<td>Detects faults in individual router wires.</td>
</tr>
<tr>
<td>pe_scan</td>
<td>Tests the serial scan chains on the ACU, PE array, and router boards.</td>
</tr>
<tr>
<td>pe_xnet</td>
<td>Exhaustive test of all three Xnet instructions. If a failure occurs, it identifies the fault's board, chip, pin number, and backplane pin number.</td>
</tr>
<tr>
<td>rts</td>
<td>Tests the transmitting, receiving, and intermediate router functions. To run rts, rtscfg must be in the same directory.</td>
</tr>
<tr>
<td>rts13</td>
<td>Tests the transmitting and receiving router functions.</td>
</tr>
<tr>
<td>rts2</td>
<td>Tests the intermediate router functions.</td>
</tr>
</tbody>
</table>

**NOTE:** The *dpumanager* executes the DPU background diagnostics about once every ten seconds when the DPU is idle. Errors found by the background diagnostics are logged in the `/usr/adm/dpujobmgr.log`.

Table 4-7  Diagnostic Tests, Sheet 2 of 2 Sheets
All diagnostic programs scroll output to the screen and copy it to the LOG file in the current directory (directory in which you executed diagnostics). If a LOG file already exists, diagnostic output is appended to it.

NOTE:
If you do not have write-permission in the directory from which you are running the diagnostics, the output goes only to the screen.

Most tests produce more than a screenful of output, and you might want to edit the LOG. The LOG files can become quite large, especially in the default verbose mode. The grep utility is useful for displaying keywords and obtaining a quick summary. For example, typing

```
# grep FAILED LOG
```
	hen then pressing Return produces a list of all the failure messages.

Standard messages show PASS or FAIL on each test and a summary of the whole diagnostic. Terse messages show only a summary of the whole diagnostic. LOG output and contents are similar to the following examples. LOG file messages are described below.

**Example of Standard Messages:**

```
Start of diagnostic test: ACU HARDWARE VME REGISTERS TEST (acu_reg1)  
Timestamp: Mon May  6 09:43:43 1991  

CUSTOMER ENVIRONMENT:  
    Standard length tests  
    Standard messages  

$Revision: 1.5 $  
PASSED.... ECSR register test: 0 errors  
PASSED.... Page Table registers test: 0 errors  
FAILED.... Instruction Memory register test: 1 error  
PASSED.... Master Interrupt Registers Test: 0 errors  
FAILED.... Instruction DMA Registers test: 2 errors  
PASSED.... Microcode Registers Test: 0 errors  
PASSED.... Serial Scan Registers test: 0 errors  
PASSED.... MAPCSR Register Test: 0 errors  
SUMMARY.... ACU HARDWARE VME REGISTERS TEST (acu_reg1).  
7 tests run, 3 errors, 0 aborts 

End of diagnostic test: ACU HARDWARE VME REGISTERS TEST (acu_reg1)  
Timestamp: Mon May  6 09:43:46 1991  
```
Example of Terse Messages:

Start of diagnostic test: ACU HARDWARE VME REGISTERS TEST (acu_reg1)
Timestamp: Mon May  6 09:44:06 1991

CUSTOMER ENVIRONMENT:
Standard length tests
Terse messages

$Revision: 1.5 $
SUMMARY... ACU HARDWARE VME REGISTERS TEST (acu_reg1).
8 tests run, 3 errors, 0 aborts

End of diagnostic test: ACU HARDWARE VME REGISTERS TEST (acu_reg1)
Timestamp: Mon May  6 09:44:09 1991
Chapter 5

Field Replacement Procedures

This chapter describes how to remove and replace the following DPU components:

- PCBs: ACU, front-end VME interface, PE array, and router
- Power trays
- Fan trays
Card Cage Access

The ACU, PE array, front-end VME interface, and router PCBs are accessed from the rear of the DPU. Figure 5-1 shows the MP 1200 card cage access; the MP 1100 has similar access.

![Diagram of Card Cage Access]

**Figure 5-1  Back of MP 1200 System and Card Cage**

To open the DPU rear door, use the 9/16" hex wrench supplied with the system.
DPU Card Cage Slots

The DPU card cage is divided into two dedicated blocks: one for I/O boards and the other for PE array boards, as shown in Figure 5-2.

MP 1100 Series systems support the ACU, the VME interface board, 4 PE array boards, and 5 I/O boards. MP 1200 Series systems support the ACU, the VME interface board, 16 PE array boards, and 15 I/O boards.

Figure 5-2   DPU Card Cage Slots
I/O Slots

The first I/O slot always holds the ACU board. The second I/O slot is configured to support an alternate VMEbus arbiter. With a DECstation 5000 front-end, the alternate VMEbus arbiter is the MVIB PCB. The other slots are for I/O boards or VMEbus boards on a MasPar® VMEbus adapter.

PE Array Board Slots

MP 1100 Series systems support 1, 2, or 4 PE array boards. MP 1200 Series systems support 1, 2, 4, 8, or 16 PE array boards. The PE array boards must be in contiguous slots, starting with the first PE array board slot. If the number of PE array boards changes, the system must be rejumerped, as described in Chapter 6.

All PE array board slots that do NOT contain a PE array board must contain a router board. The MP-1 will not work if any PE array board slot is empty.
Replacing DPU Card Cage PCBs

**WARNING:** To avoid injury or damage to equipment, make sure the system is powered down before replacing or adding any PC boards.

**CAUTION:** To avoid damage, always handle a PCB correctly:

- Handle it gently by the edges. Never handle a PCB by the top or bottom surfaces.

- Store PC boards in a static-proof container.

- Wear antistatic equipment (for instance, a wrist or ankle strap) when handling the PCB.

- If working on a PCB, use a static-free work surface.

Each PCB in the DPU card cage has ejector levers on the top and bottom of the board faceplate, shown in Figure 5-3.

Always use the ejector levers when removing or replacing PCBs:

- To release the PCB, lift up the inner end of the top lever, and press down on the inner end of the bottom lever, moving the board partway out of its slot.

- To secure the PCB in the card cage slot, press down both the inner ends of the top and bottom levers.

![Ejector Levers](image)

**Figure 5-3** PCB Ejector Levers
Replacing the ACU

The ACU board is in the first I/O slot in the DPU card cage.

To replace the ACU board, follow the steps below.

**CAUTION:** Make sure you are grounded before handling the ACU.
1. Power-down the system, as described in Chapter 3.
2. Open the DPU rear door.
3. On older models, loosen the captive screws at the top and bottom of the board faceplate. Otherwise, remove the retaining bars.
4. Use the ejector levers shown in Figure 5-3 to release the board and move it outward from its slot.
5. Carefully remove the board from its slot, handling it by the edges, and place it in a static-proof container.
6. Verify the jumper configuration on the new ACU board, as described below.
7. Carefully slide the replacement board into the slot.
8. With the ejector levers open (out), seat the board, and push the levers closed.
   **CAUTION:** Although you might have to push firmly to seat the board, do not force it.
9. Secure the board in the card cage.
10. Either tighten the captive screws on the top and bottom of faceplate or replace the retaining bars.
11. Close and latch the DPU rear door.
12. Power-up the DPU, as described in Chapter 3.
ACU Jumper Settings

The ACU jumpers provide the following functions:

- Set the ACU VMEbus address.
- Set the ACU VMEbus interrupt request level.
- Set the ACU VMEbus interrupt acknowledge level.
- Select an independent scan path from the DECstation (for diagnostics only).
- Select the ACU as a VMEbus arbiter.
- Select either 16-bit or 32-bit words for the ACU instruction DMA.
- Select the ACU instruction DMA in either block or pipelined mode.
- Set the IMem size.
- Enable and disable the ACU board clock.
- Select the carry bit.

Figure 5-4 shows the location of the ACU jumpers. The darkened blocks indicate installed jumpers.

ACU Jumper Synopsis

The default values for the ACU jumpers are as follows:

- ACU VMEbus address = 0xFCC00000
- ACU VMEbus interrupt level 1 selected
- ACU VMEbus arbiter selected
- OFF scan path selected
- Instruction DMA in pipelined mode
- 32-bit word
- Clock enabled
- IMem size = 1 MByte
Figure 5-4 ACU Jumpers

Note: Darkened jumpers are INSTALLED.
Replacing the MVIB PCB

To replace the MVIB board, follow the steps below.

1. Power-down the system, as described in Chapter 3.
2. Open the DPU rear door.
3. Disconnect the 100-pin AMP connector on the board faceplate.
4. For older models, loosen the captive screws at the top and bottom of the board faceplate. Otherwise, remove the retaining bars.
5. Use the ejector levers shown in Figure 5-3 to release the board and move it outward from its slot.
6. Carefully remove the board from its slot, handling it by the edges, and place it in a static-proof container.
7. Carefully slide the replacement board into the slot.
8. With the ejector levers open (out), seat the board, and push the levers closed.
   
   **CAUTION:** Although you might have to push firmly to seat the board, do not force it.
9. Secure the board in the card cage.
10. Either tighten the captive screws on the top and bottom of the faceplate or replace the retaining bars.
11. Reconnect the 100-pin AMP connector to the front of the board.
12. Close and latch the DPU rear door.
13. Power-up the DPU, as described in Chapter 3.
Replacing PE Array and Router PCBs

To replace PE array or router PCBs, follow the steps below.

1. Power-down the system, as described in Chapter 3.

2. Open the DPU rear door.

3. For older models, loosen the captive screws at the top and bottom of the board faceplate. Otherwise, remove the retainer bars.

4. Use the ejector levers shown in Figure 5-3 to release the board and move it outward from its slot.

5. Carefully remove the board from its slot, handling it by the edges, and place it in a static-proof container.

6. Carefully slide the replacement board into the slot.

7. With the ejector levers open (out), seat the board, and push the levers closed.

   **CAUTION:** Although you might have to push firmly to seat the board, do not force it.

8. Secure the board in the card cage.

9. Either tighten the captive screws on the top and bottom of faceplate or replace the retainer bars.

10. Close and latch the DPU rear door.

11. From the front, open the DPU front and inner front doors, and make sure the DPU backplane jumpers are installed correctly, as described in Chapter 6.

12. Close the DPU front doors.

13. Power-up the DPU, as described in Chapter 3.
Replacing DPU Power Trays

**WARNING:** To avoid injury or damage to the equipment, remove all jewelry before working on power supplies.

**MP 1200 Power Tray**

Follow the steps below to remove and replace the MP 1200 power tray.

**Removal**

**NOTE:** As necessary, label all cables when you disconnect them.

1. Shut down the DECstation, and power-down the workstation and the MP 1200, as described in Chapter 1.

2. Set the DPU 30A breaker switch OFF (down), and unplug the power cord from the wall outlet.

3. Open the back door of the MP 1200.

4. To avoid possible problems when powering up, use the ejector levers shown in Figure 5-3 to move each PCB from the backplane, in both the upper and lower card cages.

5. Disconnect all the cables to the back of the power tray.

6. Open the DPU front door.

7. Remove the DPU inner front door.

8. Using two 9/16" open-end wrenches, remove the +5V positive and negative power bars. Note the offset on the negative post, which is needed for clearance between the positive and negative posts.

**CAUTION:** To avoid damage to the power post on the −12V supply, do not use a deep socket wrench when removing the positive +5 bar.

9. Using a 9/16" wrench, remove the cables from the −5V supply.

10. Using a 5/16" wrench, remove the +12- and −12V cables.

11. On older power trays, remove the ground cable running to a screw in the top center of the power tray. Loosen the screw and slide the ground wire out.

12. Cut any cable restraints fastening cables to the power tray.

13. Remove the four Phillips screws attaching the power tray to the chassis.

14. Make sure you have disconnected ALL cables from the back of the power tray.

**CAUTION:** Do not let the power cables get caught on the power tray while sliding it out or in. Cut cable insulation can cause a direct short.
15. From the DPU front, gently slide the power tray forward, checking that no cables are getting caught on the power tray.

Make sure the attached AC power cord does not get caught coming through the lower rear door.

Do not place any weight on the back of the power tray.

Replacement

1. Gently slide the new power tray into place, feeding the AC power cord through the DPU lower rear door.

   **CAUTION:** Before sliding the tray in the last few inches, make sure there are no cables in the back of the machine blocking its way.

2. Replace the four Phillips screws fastening the power tray to the frame.

3. Connect the +12V and −12V cables to the power supply.

   **CAUTION:** To avoid snapping the power post, apply minimum force when tightening the nut.

4. Connect the ground wire to the top of the power tray. (only on older MP 1200s)

5. Install the −5V cables. The cable going to the top post (+) is actually ground if you failed to mark the cable earlier.

6. Install the +5V power bars. It is easiest to install the positive bar first, then the negative bar.

7. Make sure all cables are connected properly and there are no shorts.

8. Power on the DPU in local mode.

9. Measure voltage levels, and if needed, adjust levels on the power supplies.

   Measure the voltages on the backplane. Voltage settings are described in Chapter .

10. Power down the DPU.

11. Using the ejector levers, reinstall all the card cage PCBs, seating them firmly against the backplane.

12. Power on the DPU in LOCAL mode.

13. Replace the DPU front inner door, and close the front door.

14. Power-up the DECstation.

15. Set the DPU 30A or 15A circuit breaker ON (up).

16. On the backplane, remeasure the +5V, −5V, and 12V levels, and adjust if necessary.

17. Run diagnostics or applications to verify the system operation.
Figure 5-5  Power Tray Rear Panel
MP 1100 Power Tray

Currently, the MP 1100 uses two types of power supplies: Powertec and HC Power. The power supplies are interchangeable, although the connections are slightly different. Refer to Figure 5-6 and specific directions below when replacing these power supplies.

**WARNING:** To avoid injury or damage to equipment, remove all jewelry before working on power supplies.

**NOTE:** As necessary, label *all* cables and connectors as you remove them.

**Removal**

1. Shut down the DECstation, and power-down the workstation and the MP 1100, as described in Chapter 1.
2. Set the DPU 15A breaker switch OFF (down), and unplug the power cord from the wall outlet.
3. Open the DPU back door.
4. To avoid possible problems when powering up, use the ejector levers shown in Figure 5-3 to detach each PCB from the backplane.
5. Disconnect all cables to the back of the power tray.
6. Open the DPU front door, and remove the front inner door.
7. Using a 9/16" open-end wrench or socket wrench, remove the cables from the +5V power supply.
8. Remove -5V, +12V, and -12V wires from power supplies:
   - **Powertec** Using the 5/16" socket wrench, remove the -5V, +12V, and -12V wires.
   - **HC Power** Using a Phillips screwdriver, remove the -5V, +12V, and -12V wires.
9. Remove the four Phillips screws holding the power supply unit to the frame.
10. Make sure you have removed ALL cables from the back of the power tray.

**CAUTION:** Do not let the power cables get caught on the power tray while sliding it out or in. Cut cable insulation can cause a direct short.

11. From the DPU front, carefully lift the power supply cables out of the way while pulling the power supply unit out of the frame.
12. Lay the power supply unit down flat or on its side.

Do not put weight on the back of the tray.
Caution: Over tightening the 12V wires can break the posts on the power supply.

Figure 5-6 MP 1100 Power Supplies
Replacement

1. Slide the new power tray into place.

2. Lift the power supply cables out of the way, and slide in the power tray.

3. Replace the four Phillips screws fastening the power supply unit to the frame.

4. Replace -5V, -12V, and +12V wires:

   **Powertec power supply:**

   Use a 5/16 socket to secure the nuts after the wires are in place.

   **CAUTION:** To avoid snapping the power post, apply minimum force when tightening the nut.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Value</th>
<th>Wire</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5V</td>
<td>+</td>
<td>Black</td>
<td>Ground from backplane</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Red</td>
<td>-5V to backplane</td>
</tr>
<tr>
<td>-12V</td>
<td>+</td>
<td>Black</td>
<td>Jumper to +12V (-) terminal</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>White</td>
<td>-12V to backplane</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Blue</td>
<td>Ground for fan tray</td>
</tr>
<tr>
<td>+12V</td>
<td>+</td>
<td>Red</td>
<td>+12V to backplane</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td>Red</td>
<td>To fan tray</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Black</td>
<td>Ground to backplane for +/- 12V</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Black</td>
<td>Jumper to -12V (+) terminal</td>
</tr>
</tbody>
</table>
HC Power supply:
Use a Phillips screwdriver to secure the wires to the power supply, referring to Figure 5-6:

\[
\begin{align*}
V1 &= +5V \\
V2 &= -5V \\
V3 &= +12V \\
V4 &= -12V \\
\end{align*}
\]

- **+V2**  Black  Ground from backplane
- **-V2**  Red  -5V to backplane
- **+V3S**  -----  Blank
- **+V3**  Red  +12V to backplane
- **Red**  To fan tray
- **-V3**  Black  Ground to backplane for +/- 12V
- **Black**  Jumper to +V4
- **-V3S**  -----  Blank
- **+V4**  Black  Jumper from -V3
- **-V4**  White  -12V to backplane
- **Blue**  To fan tray
- **+V5**  -----  Blank
- **-V5**  -----  Blank

5. Connect the +5V cables.
6. Connect the cables on the back of the power supply assembly.
7. Before powering-on the system, make sure all cables have been installed properly and that there are no short circuits.
8. Power-on the DPU in LOCAL mode.
9. Measure voltage levels, and if needed, adjust levels on the power supplies.
   Measure the voltages on the backplane. Voltage settings are described in Chapter.
10. Power-down the DPU.
11. Using the ejector levers, replace the card cage PCBs, seating them firmly against the backplane.
12. Power-on the DPU.
13. On the backplane, measure the power supply voltages levels again and adjust, if necessary.
14. Close the DPU doors.
15. Run diagnostics to verify proper performance.
Replacing DPU Fan Tray

Follow the steps below to replace the DPU fan trays.

**MP 1200 Fan Tray**

1. Power-down the MP 1200 DPU, set the 30A circuit breaker OFF (down), and disconnect the power cord from the wall outlet.

2. Remove the front door and the front inside door.

3. From the back of the DPU, unplug the power cable from the power supply to the fan tray.

4. Using a 5/16" open-end wrench and a 5/16" socket wrench, remove the +5V bus bars from the +5V power supply. Gently pull them forward, and lay them down so that they are out of the way of the fan tray when it is pulled out.

5. Remove the four Phillips screws fastening the fan tray to the frame.

6. Pull the fan tray out smoothly.

   **CAUTION:** Be careful not to pinch the +5V cables. This can break the insulation, causing the +5V power to short to ground.

7. Before installing the new fan tray, note the arrow indicating the tray’s proper orientation.

   Slide the new fan tray in.

   **CAUTION:** Be careful not to pinch the +5V cables. This can break the insulation, causing the +5V power to short to ground.

8. Replace the four Phillips screws securing the fan tray to the frame.

9. Connect the power cable to the fan tray.

10. Connect the +5V bus bars. Make sure the bus bars are not shorting to the frame or to each other.

11. Power on the machine, and verify the new fan is operating properly.
MP 1100 Fan Tray

Follow the steps below to remove and replace the MP 1100 fan tray.

1. Power down the system, as described in Chapter 1, set the 15A circuit breaker OFF (down), and unplug the power cord from the wall outlet.

2. Open the front and lower front doors.

3. Open the front inner door.

4. Remove the front inner door.
   - Unplug the switch connector from the lightpipe (under the top cover).
   - Pull down on the latch on the upper hinge and pull the door out.
     Place the door upright against a wall.

5. Disconnect the power connector from the fan tray.

6. Remove the four Phillips screws securing the fan tray to the frame.
   In some cases, to be able to slide the fan tray out, you will need to remove the +5V cables from the power supply.
   Remove those cables now if necessary.

7. Slide out the fan tray.

8. Slide in the replacement fan tray.
   If previously removed, reconnect the +5V cables to the power supply.

9. Replace the four Phillips screws securing the fan tray to the frame.

10. Connect the power connector to the fan tray.

11. Replace the front inner door:
    - While holding the upper hinge latch down, put the door in place, securing the lower hinge and then sliding the upper hinge into place so that the latch connects properly.
    - Reconnect the switch connector to the rightmost lightpipe connector.

12. Close and latch the front inner door.

13. Close the front and lower front doors.

14. Power-up the DPU, and make sure the fan is working correctly.
Chapter 6

Upgrading PE Arrays

This section describes how to add additional PE array PCBs, thereby increasing system performance.

NOTE:

- Every system must have 1, 2, 4, 8, or 16 PE array PCBs. No other combinations are supported.

- When you add PE array PCBs, you must change the DPU backplane X-Net jumpers.

- PE array PCBs start in the first PE array board slot and occupy contiguous slots up to the maximum installed. All remaining PE array board slots must have a router board installed. No PE array board slot can be left empty.
System Issues for Upgrades

No system software changes are required. The system software automatically reconfigures itself to accommodate any legal number of PE array PCBs.

With one exception, DPU programs will run on any size PE array, and increasing the array size increases the run speed. However, if an MPL program is hard-coded with the size of the processor array, the program needs to be changed and recompiled before being run on a different-sized PE array.
Handling PCBs

**CAUTION:** To avoid damage, always handle a PCB correctly:
- Handle it gently by the edges. Never handle a PCB by the top or bottom surfaces.
- Store PCBs in a static-proof container.
- Wear antistatic equipment (for instance, a wrist or ankle strap) when handling the PCB.
- If working on a PCB, use a static-free work surface.

Each PCB has ejector levers on the top and bottom of the board faceplate, shown in Figure 6-1. Always use the ejector levers when removing or replacing PCBs:
- To release the PCB, lift up the inner end of the top lever, and press down on the inner end of the bottom lever, moving the board partway out of its slot.
- To secure the PCB in the card cage slot, press down both the inner ends of the top and bottom levers.

![Ejector Levers](image)

Figure 6-1 PCB Ejector Levers
Adding PE Array PCBs

To add additional PE array PCBs, follow the steps below, and refer to Figure 6-3.

1. Power-down the DPU as described in Chapter 3.

2. Open the DPU rear door to access the card cage, as shown in Figure 6-2.

3. Identify the slot(s) where you are going to install the PE array board(s), as shown in Figure 6-3.

4. Remove the router board(s) from the slot(s):
   - On older models, loosen the captive screws at the top and bottom of the board faceplate. Otherwise, remove the retaining bars.
   - Use the ejector levers to release the board and move it outward from its slot.

5. Carefully slide the new PE array board into the slot. When it is almost in place, you might encounter some resistance; push firmly to seat it in the backplane connectors.

6. Secure the board to the card cage by closing the ejector levers.

7. Either tighten the captive screws at the top and bottom of the board faceplate or replace the retaining bars.

8. Close and latch the DPU rear door.

9. Open the DPU front and inner front doors.

10. Reconfigure the backplane Xnet jumpers, as described on page 6-10.

11. Close and latch the DPU inner front and front doors.

12. Power-up the system, as described in Chapter 3.
Figure 6-2  DPU Rear Doors
MP 1100 Card Cage

MP 1200 Card Cage

Figure 6-3  DPU Card Cage Slots
DPU Backplane Jumpers

To understand the backplane jumpering, you need to understand the slot arrangement in the card cage. In both the MP 1100 and the MP 1200, the DPU card cage is divided into two regions:

- The ACU, the alternate VMEbus arbiter, and the I/O board slots
- The PE array and the router board slots

Figure 6-4 shows the location of the board slots and corresponding sets of jumpers on the MP 1200 and MP 1100 backplanes.

The MP 1200 backplane has two rows of card cage slots. The top row is for I/O slots, and the bottom row is for PE array board slots. Looking at the backplane slots from the DPU *front*, the ACU is in the upper-right I/O slot. The second I/O slot from the right is the front-end VME interface board (the alternate VME arbiter).

The bottom row has 1, 2, 4, 8, or 16 PE array boards, in contiguous locations starting from the right, and router boards in any slots not containing PE array boards.

*(NOTE: When you are looking *into* the card cage, this view is *reversed*.)*

The MP 1100 card cage has one row of 11 card cage slots. Looking at the backplane slots from the DPU *front*, and starting from the right, the first is the ACU board slot, the second is the front-end VME interface board, the next five are I/O slots, and the remaining four are PE array board slots.

The PE array slots have one, two, or four PE array boards, at contiguous locations, and router boards in any slots not holding PE array boards. As shown in Figure 6-4, the fourth slot from the left holds the first PE array board.

*(NOTE: When you are looking *into* the card cage, this view is *reversed*.)*
Figure 6-4  DPU Backplane Jumpers
Backplane Access

The backplane access is through three doors on the front of the DPU, as shown in Figure 6-5.

1. Open the outer top and bottom doors.
2. Open the inner door, using the door latch key.
3. Reverse these steps to close the DPU front doors: close the inner door first, then the bottom door, and finally the outer door.

Figure 6-5 System Front Doors
Xnet Jumpers

The Xnet jumpers control the Xnet connections between PE array boards. They are on the bottom of the backplane, beneath PE array slots 0 through 15 on MP 1200 systems and beneath PE array slots 0 through 3 on MP 1100 systems.

**CAUTION:** When changing Xnet jumpers:

- Only slots with PE array boards should have Xnet jumpers installed. Never install any Xnet jumper on a slot that does NOT contain a PE array board.

- The Xnet jumper installations for *all* occupied PE array board slots can change when the number of PE array boards in the system changes. When you increase or decrease the number of PE array boards, review the Xnet jumper configurations for *every* slot containing a PE array board.

Figure 6-6 shows the Xnet jumpering for MP 1100 configurations, and Figure 6-7 shows the Xnet jumpering for MP 1200 configurations.

**Figure 6-6** MP 1100 Xnet Jumper Configurations
Single PE-array board systems

Two PE-array board systems

Four PE-array board systems

Eight PE-array board systems

Sixteen PE-array board systems

The lines represent jumper wires.

There should be NO jumpers on J8 through J15.

Figure 6-7 MP 1200 Xnet Jumper Configurations
Upgrading PE Arrays
Appendix A

Installation Checklist

This appendix provides a checklist of the basic installation steps. For more detailed instructions, see Chapter 2.

Installation Steps

1. Make sure your site meets the requirements listed in the *MasPar MP-1 Site Prep Guide*.

2. Install the DECstation, as described in the DEC documentation.

3. Run the DEC power-up self test.

4. Install the DPU:
   - Make sure the DECstation is powered-down.
   - Make sure the DPU keyswitch (front inner door) is OFF and the rear panel 30A (or 15A) breaker is OFF (down).
   - Install DPU PCBs.
   - Connect the data cable to the DPU cable connector panel.
   - Connect the data cable to the DECstation rear panel.
   - Connect the RS-232 cable to the DPU cable connector panel.
   - Connect the RS-232 cable to the DECstation rear panel.
   - Connect the external telephone cable to DPU X11 jack, and configure the system to support the modem.
5. Make sure the DPU power selector switch (inner rear panel) is set to LOCAL.
6. Connect the DPU power cable.
7. Power-up the DECstation.
8. Power-up the DPU, turning the keyswitch ON and the 30A (or 15A) breaker ON (up).
9. Boot the DECstation.

You can now proceed with system administration.
Appendix B

Power Cable Wiring and Replacement

Power Cable Wires

MP 1200 systems shipped outside the United States are shipped with bare wires at the ends of the power cable. To conform to local wiring standards, you might need to do one of the following:

- Connect the bare wires to a locally compatible plug.
- Connect the bare wires to a terminal block.
- Replace the cable entirely.

**WARNING:** To avoid injury and damage to equipment, make sure all equipment is unplugged before working on any power cable. Also, make sure all changes are inspected by a qualified electrician familiar with local codes.

Wire Ends

If you connect the bare wire ends to a terminal block or a plug, connect green to ground, white to neutral, and black to hot (live).

If you are wiring the ends to the different phases of a 110V circuit, white and black can be wired either way.
Replacing the Power Cable

If necessary, the system power cable can be replaced to conform to local codes. Refer to Figure B-1, and follow the steps below.

**WARNING:** To avoid injury and damage to the equipment, never work on the cable wiring while any part of the system is connected to power.

1. Make sure both the VAXstation and the DPU are powered down. Set the DPU rear panel circuit breaker switch OFF (down).

2. Remove the power supply assembly, as described on page 5-11.

3. Remove the old cable:
   - Loosen the captive screws from the top of the power enclosure.
   - Remove the top cover.
   - Remove the power cable leads from the ground stud and the terminal block. Loosen the screws that are recessed into the top of the terminal block; this loosens the power leads.
   - Loosen the strain relief fitting by turning the locking nut clockwise.
   - When the strain relief fitting is loose, feed the cable through the hole and out the power enclosure.

4. Feed the new cable in through the strain relief fitting.

5. Connect the power cable ground lead to the ground stud.

6. Connect the power cable hot (live) lead to the connectors on the right side of the terminal block.
   - Loosen the screws that are recessed into the top right side of the terminal block.
   - Feed the power cable leads into the holes in the right side of the terminal block.

<table>
<thead>
<tr>
<th>Color</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>HOT</td>
</tr>
<tr>
<td>Black</td>
<td>Neutral</td>
</tr>
<tr>
<td>Green/Yellow</td>
<td>Ground</td>
</tr>
</tbody>
</table>

- If you are connecting to a 220-V circuit with hot and neutral leads, connect HOT to the near side of the terminal block and NEUTRAL to the far side. The corresponding wires on the other side of the terminal block should be RED for HOT and BLACK for NEUTRAL. *Do not change these.*
- If you are connecting to the different phases of a 110-V circuit, connect the leads to either of the connectors on the terminal block. It works properly either way.

- Tighten the screws on top of the terminal block until the power cable leads are fastened securely.

7. Tighten the strain relief fitting by turning the lock nut.

**NOTE:** Before you tighten the strain relief fitting, feed some extra cable into the power enclosure, so there is no strain on the terminal block connections.

8. Replace the top cover.
Figure B-1  Power Cable Wiring
Index

15A circuit switch 3-3
1A breaker switch 3-3

2500 telephone circuit 2-20

30A circuit switch 3-3

Accessing the card cage 2-13, 5-2
ACU 1-8
crystal oscillator 2-15
installation 2-15
interrupt level 5-7
jumper settings diagram 2-11
jumpers 2-10, 5-7
overview 1-12
removal and replacement 5-6
VMEbus address 2-10, 5-7
VMEbus interrupt level 2-10, 5-7
ACU-PE Bus communications 1-16
acu_bound 4-18
acu_clim 4-18
acu_diag 4-16, 4-18
acu_int 4-18
acu_int2 4-18
acu_macro 4-18
acu_micro 4-18
acu_pgtbl 4-18
acu_pptest 4-18
acu_prof 4-18
acu_reg1 4-18
acu_reg2 4-18
acu_sup 4-18

Adding PE array PCBs 6-1
Arbiter, VMEbus 2-14, 5-4
Array Control Unit
See also ACU
Asynchronous VME Interface PCB
See also AVME PCB
AVME PCB 1-8

Backplane
voltage measurement 3-6
Backplane jumpers 6-7
Breaker switch, 15A 3-3
Breaker switch, 1A 3-3
Breaker switch, 30A 3-3

Cable panel, DECstation 2-19
Card cage 1-8
Card cage access 2-13, 5-2
Card cage slot assignments 5-3
Card cage, accessing 5-2
Checklist, installation A-1
Circuit breakers, DPU 3-3
Code changes and PE array size 6-2
Connecting power cable 2-21
Connecting the modem 2-20
Container inspection 2-2

Damage to shipping container 2-2
DECstation
cable panel 2-19
self-test messages 2-8
unpacking instructions 2-8
Diagnostics 4-14
environment 4-14
LOG file 4-20
options 4-14
order 4-16
output 4-14
quick mode 4-14
running 4-14
suspending 4-14, 4-15
terse mode 4-14
types of 4-16

DPU
- card cage 1-8
circuit breakers 3-3
connecting power cable 2-19, 2-21
control switches 3-1
inner door 6-9
installation 2-9
jumpers 2-16
keyschitch 3-3
overview 1-6
power selector 3-3
power tray replacing 5-11
unpacking instructions 2-3

Ejector levers 2-9, 2-10, 5-5, 6-3
Enclosure
MP 1100 5-3
MP 1200 5-3
/etc/uerf 4-8
Exterior panel figure, DECstation 2-19, 3-11

Fault Code Word 4-8
Field replacement procedures 5-1
fltcod 4-10
Front-end system 1-5, 4-14
Front-end VME interface 2-14, 5-4

Global router overview 1-19

Hardware installation 2-8
checklist A-1

I/O Channel Controller PCB
See also IOCTL PCB
I/O RAM PCB
See also IORAM
I/O slots 2-14, 5-4
Inner door, DPU 6-9
Installation
ACU board 2-15

checklist A-1
DECstation 2-8
DPU 2-9
hardware 2-8
PCBs 2-15
Interrupt level, ACU 2-10, 5-7
IOCTL PCB 1-8
IORAM 1-11
IORAM PCB 1-8

Jumpers
ACU 2-10, 5-7
backplane 6-7
DPU 2-16
Xnet 6-10

Keyschitch, DPU 3-3

LED, power status 3-4
LEDs
PE array boards 4-6
router boards 4-6
./LOG 4-8, 4-20
LOG file 4-20
interpreting 4-20
location 4-20
messages 4-15

Models, MP-1 1-3
Modem 2-20
MP 1100 1-3
enclosure 5-3
MP 1200 1-3
enclosure 5-3
power cable B-4
wiring B-4
MP-1
models 1-3
overview 1-1
MPIOC 1-11
$MP_PATH/field/bin 4-14
MVIB PCB 1-8, 1-11

Number of PE array boards 5-4

Overview, MP-1 1-1

Parallel VME PCB
See also PVME PCB
PCB
ejector levers 2-10, 5-5, 6-3
PE array
overview 1-13
slots 2-14, 5-4
upgrades 6-1
PE array boards
adding 6-1
LEDs 4-6
number of 5-3, 5-4
PE array size
affect on code 6-2
PE communications 1-16
PE PCB 1-8, 1-12
PE Private Memory
See also PMEM
PE Private Register
See also PREG
PE Special Purpose Register
See also SREG
pe_arith 4-19
pe_block 4-19
pe_diag 4-16, 4-17, 4-19
pe_func 4-19
pe_macro 4-19
pe_memdiag 4-19
pe_micro 4-19
pe_rtpb 4-19
pe_rtdiag 4-19
pe_rtr 4-19
pe_rtxcnt 4-19
pe_scan 4-19
pe_xnet 4-19
PMEM 1-13
PMSTAT 4-11
Power cable replacement B-4
Power cable, MP 1200 B-4
Power requirements 2-19, 2-21
Power selector, DPU 3-3
Power status LED 3-4
Power supply
adjustment 3-6
wiring 3-6
Power tray replacement
MP 1100 5-14
MP 1200 5-11
Power-down sequence 3-11, 3-12
Power-up sequence 3-11
Power-up sequencer 3-4
Precautions, static discharge 2-9, 5-5, 6-3
PREG 1-13
Private RAM
See also PMEM
Private Registers
See also PREG
Processor Element PCB
See also PE PCB
PVME PCB 1-8, 1-10
Replacement procedures
ACU board 5-6
card cage PCBs 5-5
MP 1200 power tray 5-11
MP 1100 power tray 5-16
MP 1200 power tray 5-11
MVIB PCB 5-9
PE array PCB 5-10
power cable B-4
router PCB 5-10
Router chip 1-12, 1-19
Router overview 1-19
Router PCB 1-8, 1-12
LEDs 4-6
rts 4-19
rtsl3 4-19
rtst 4-19
Running a diagnostic test 4-14
Self-test messages, DECstation 2-8
Sequencer, power-up 3-4
Shipping container damage 2-2
Shockwatch 2-2
Slot assignments, card cage 5-3
Special Purpose Registers
See also SREG
SREG 1-13
SREG0 1-13
Static discharge precautions 2-9, 5-5, 6-3
Suspending a diagnostic 4-14
SVME PCB 1-8
Synchronous VME Interface PCB
See also SVME PCB
System power requirements 2-19, 2-21
System power-down 3-11, 3-12
System power-up sequence 3-11
Telephone connection 2-20
Terse mode 4-14
Test suites 4-16
Tip-n-Tell 2-2
Unpacking
diagrams 2-6, 2-7
instructions, DECstation 2-8
instructions, DPU 2-3
/usr/adm/dpujobmgr.log 4-8
VMEbus 5-3
VMEbus address, ACU 2-10, 5-7
VMEbus arbiter slot 2-14, 5-4
Voltage measurement 3-6

Wiring, MP 1200 B-4

X-Net communications 1-18
Xnet jumpers 6-10
This manual describes system administration for the MasPar MP-1 with V3.0 software and a DECstation 5000 front end.
MasPar Computer Corporation makes no representation or warranties regarding the contents of this document and reserves the right to revise this document or make changes in the specifications of the product described herein at any time without obligation to notify any person of such revision or change.

Copyright © 1990, 1991, 1992 MasPar Computer Corporation, Sunnyvale, California

All Rights Reserved
Printed in the United States of America

This manual, its format, and the hardware, software, and microcode described in it are copyrighted by MasPar Computer Corporation and may not be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the express written permission of MasPar Computer Corporation.

Restricted Rights Legend:
The software and microcode documented in this manual has been developed at private expense and is not in the public domain. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

MasPar and the MasPar logo are registered trademarks of MasPar Computer Corporation. MasPar Programming Environment, MasPar Fortran, MasPar Programming Language, MasPar Data Display Library, MasPar Image Processing Library, MasPar Mathematics Library, and MasPar Parallel Disk Array are trademarks of MasPar Computer Corporation.
DEC, DEC AVS, DECnet, DECwindows, DECstation, ULTRIX, VAX, and VAXstation are trademarks of Digital Equipment Corporation.
UNIX is a registered trademark of the American Telephone and Telegraph Company.
X Window System is a trademark of the Massachusetts Institute of Technology.
Motif is a trademark of the Open Software Foundation, Inc.
NFS is a trademark of Sun Microsystems.
AVS is a trademark of Advanced Visual Systems, Inc.
PostScript is a registered trademark of Adobe Systems Incorporated.
Sun, SunOS, Sun-4, SPARC, and SPARCstation are trademarks of Sun Microsystems, Inc.

MasPar Computer Corporation
749 North Mary Avenue
Sunnyvale, California 94086
1-800-526-0916
1-408-736-9560 (FAX)
Preface

This manual describes system administration procedures specific to the MasPar® system. The MP-1 system consists of a front end DECstation 5000 running ULTRIX, the MasPar Data Parallel Unit (DPU), MasPar system and product software, and attached (optional) devices.

This manual is for system administrators of the MasPar system. This manual covers only those aspects of system administration that apply to the DPU, its interface to your DECstation, and the optional hardware devices available with the MasPar system. Refer to your ULTRIX documentation for details on basic UNIX system administration.

This manual assumes you are already familiar with standard system administration procedures for ULTRIX, the DECstation 5000, and your window manager.
How this Manual is Organized

Chapter 1 introduces the MP-1 and basic system administration.

Chapter 2 discusses how to install the MasPar software and software updates.

Chapter 3 discusses how to configure and update the MasPar software. It also discusses how to set up new users, and how to set up remote access to the MasPar software.

Chapter 4 contains information on the MasPar file systems and how to mount and administer them.

Chapter 5 describes MasPar daemons and utilities, including the DPU manager and the license manager.

Chapter 6 describes the system administration information for the MPDA (MasPar Parallel Disk Array).

Chapter 7 describes how to avoid and solve common problems that you or MP-1 system users might encounter.

Help

To get general information about MasPar, including how to contact MasPar, type "man maspar" at the shell prompt.

To get on-line help on MasPar commands, compilers, and library routines, type "man -k maspar" or "apropos maspar" at the shell prompt to see what MasPar man pages are available.

To get more general help on how to program the MasPar parallel processing system, study the examples in $MP\_PATH/examples.

If you have additional questions, or if you want to report defects or make comments on MasPar products and documentation, you can contact MasPar via electronic mail, telephone, or FAX at the following locations:
Typographic Conventions

The following conventions are used throughout this book:

This is used: text in *italics* To show: program variables, new terms

This is used: text in **typewriter font** To show: names of items the user selects, text that appears on screen, examples, path names, command names

This is used: text in **typewriter bold font** To show: what you need to type, such as filenames. When the text begins with a #, it means you must be logged in as user "root" to enter the commands shown. When it begins with a %, it means you must be logged in as user "maspar."

This is used: text in **typewriter bold italic font** To show: variables (such as program names) that the user needs to supply

This is used: Initial Caps on text To show: names of tools, windows, panes, and menus

This is used: "text in quotation marks" To show: literals

For More Information

You may want to refer to the following manuals for more information.

For: Background information on the MasPar MP-1 system and its architecture

See: *MasPar System Overview.* PN 9300-0100.

For: Information on MasPar system commands


For: *MasPar Commands.* PN 9300-0300.
For:

For details on installing and servicing the MasPar machine

See:

MP-1 Installation and Service Manual
PN 9300-9023.

Installation Guide for MasPar System Software, V3.0
PN 9300-9025.

Site Prep Guide
PN 9305-5000.

For complete hardware service information for the MasPar Parallel Disk Array (MPDA)

MasPar 3000-Series Parallel Disk Array Manual
PN 9300-5005

MasPar 4000-Series Parallel Disk Array Manual
PN 9300-9040

Information on how to use MPPE

MPPE User Guide.
PN 9305-0000.

MPPE Quick Reference.
PN 9305-0300.

Information on MasPar’s optional libraries

PN 9302-0200.

PN 9302-0300.

PN 9302-0400.

Information on MasPar’s programming languages and compilers

MPL User Guide.
(ANSI C version)
PN 9302-0101.

(ANSI C version)
PN 9302-0001.

MasPar Fortran User Guide
PN 9303-0100.

General UNIX system administration information

UNIX® System Administration Handbook
Nemeth, Snyder, and Seebass
Prentice Hall Software Series

UNIX System Manager's Manual
4.3 Berkeley Software Distribution for Virtual VAX-11 Version,
University of California,
Berkeley, California.
MasPar system software Version V3.0 is based on ULTRIX 4.2a. For the DECstation, the ULTRIX document set from DEC provides the standard system administration information.

Complete information on administration of ULTRIX and your DECstation

See:
ULTRIX-32 general information, volume 1, reader’s guide, and master index.
Digital Equipment Corporation. 1988
Order number AA-ME82A-TE.

ULTRIX worksystem software reference pages.
Digital Equipment Corporation. 1988
Order number AA-MA85A-TE.

ULTRIX worksystem software advanced installation guide.
Digital Equipment Corporation. 1988
Order number AA-KU43B-TE.

ULTRIX worksystem software DECwindows user’s guide.
Digital Equipment Corporation. 1988
Order number AA-MA87A-TE.

You may also want to refer to the documentation for your window manager.
# Table of Contents

## Preface
- How this Manual is Organized .................................................. ii
- Help ....................................................................................... ii
- Typographic Conventions ......................................................... iii
- For More Information .............................................................. iii

## Chapter 1. MasPar System Administration 1-1
- X Window System Compatibility ................................................ 1-2
- The MasPar Directory Tree ....................................................... 1-2
  * Maintaining Multiple MasPar Directory Trees ......................... 1-3
- User `maspar` and Group `maspar` ............................................. 1-3

## Chapter 2. Installing and Re-installing Software 2-1
- Installing ULTRIX 4.2a Software from Scratch ......................... 2-2
- Installing the ULTRIX Software Upgrade ................................... 2-5
- Installing V3.0 MasPar System Software ................................... 2-6
- Upgrading to V3.0 MasPar Software ......................................... 2-9

## Chapter 3. Configuring Your Installation 3-1
- Configuring For a Network Environment .................................... 3-2
- Configuring For a Stand-Alone Environment .............................. 3-2
Setting Up and Using the Modem ........................................ 3-3
  Configuring Software for the Modem ................................ 3-3
  Using the Modem .................................................... 3-4
Setting Up the Motif Window Manager ............................ 3-4
Setting Up Your Environment ....................................... 3-5
Adding New Users .................................................... 3-6
Remote Setup .......................................................... 3-7
  Running on a Remote DECstation ................................. 3-7
  Running on a SPARCstation ....................................... 3-8
  Updating a Host Name ............................................. 3-9
Yellow Pages Setup .................................................. 3-10

Chapter 4.  File System Management 4-1
NFS Exporting of MPFS and IORAM File Systems ............... 4-2
VMEFS file system ................................................... 4-2
The IORAM File System ............................................ 4-4

Chapter 5.  Utilities and Daemons 5-1
Working with the License Manager ............................... 5-2
  License Manager Files ........................................... 5-2
    license.dat ..................................................... 5-2
  Updating a license.dat File ..................................... 5-3
  Conflicts With ULTRIX ........................................... 5-3
Conflicting Licenses ................................................ 5-4
  Restoring a license.dat File .................................... 5-4
  When License Limits Are Exceeded ............................ 5-5
License Manager Utilities ........................................... 5-5
The maspar Daemon .................................................. 5-6
Using dpumanager ..................................................... 5-6
  Restarting dpumanager ......................................... 5-6
  Swap-to-Disk and the dpumanager .............................. 5-7
Setting dpumanager Options ....................................... 5-7
    Modifying dpumanager with mpctl .......................... 5-8
    Understanding Swapping Conditions ....................... 5-9
    Swap Scheduling ............................................. 5-10
    Special Notes ................................................. 5-12
    Swap File ..................................................... 5-12
Other MasPar Utilities ............................................. 5-12

Chapter 6.  MPDA System Administration 6-1
Operating System Configuration .................................. 6-2
  Establishing MPDA Communications ............................. 6-3
  No Device Entry .................................................. 6-3
Chapter 7. Troubleshooting Common Problems

Problems with DPU Access ........................................... 7-2
Accessing the DPU .................................................. 7-2
Determining if the DPU Is Running ......................... 7-2
Starting the Job Manager ......................................... 7-3
Checking the DPU Connections ............................... 7-3
Reporting Problems .................................................. 7-3

License Manager Problems ........................................ 7-4
License Limits ...................................................... 7-4
License Failures .................................................... 7-4
Conflicting Licenses ............................................... 7-4
Reporting Problems ............................................... 7-4

Compiler Problems ............................................... 7-5
Logging Compiler Activity ...................................... 7-5
Reporting Compilation and Execution Problems ........... 7-6

MPPE Problems ....................................................... 7-6
Incorrect DISPLAY Variable ..................................... 7-6
Incorrect Host Name .............................................. 7-6
Incorrect User ID .................................................. 7-6
Incorrect System Date ............................................. 7-7
Reporting MPPE Problems ....................................... 7-7

Troubleshooting the Swap To Disk Facility .................. 7-7

Index
Table of Contents
Chapter 1

MasPar System Administration

This manual describes system administration for the MasPar® MP-1. MasPar system administration consists of working with the DPU and ULTRIX to manage the DPU, to configure new installations, and to install software upgrades.

The MP-1 consists of:

• a Data Parallel Unit (DPU). The DPU executes those parts of MasPar programs that require parallel processing. The DPU has a small kernel, called the ACU kernel, that performs services for the control software that runs under ULTRIX. The ACU kernel and the ULTRIX kernel communicate directly via interrupts and control registers.

• a front end DECstation 5000 running ULTRIX, DEC's version of the UNIX operating system. MasPar System Software version 3.0 is based on ULTRIX 4.2a.

MasPar has added some utilities to the standard ULTRIX distribution to enable you to administer the DPU; these are described in Chapter 5.
X Window System Compatibility

MasPar system software Version V3.0 includes DECwindows revision 4.2a. This includes the

The OSF Motif™ window manager is distributed with V3.0 software for use with MPPE, the
MasPar Programming Environment. MasPar recommends that you use Motif as your window
manager. MPPE will operate properly with any X windowing system, however its windows and
widgets will work and look like Motif windows and widgets no matter which windowing system
you use.

The MasPar Directory Tree

Most of the MasPar software distribution is located in the directory /usr/maspar. The
/usr/maspar tree contains these directories and files:

- **avs**: Directory that contains the AVS user interface for MPPE. Note that AVS is
  an optional product; MasPar only provides the interface to AVS, not the AVS
  product.
- **bin**: Directory that contains executables, such as compiler drivers, for use by all
  general users.
- **demos**: Directory that contains the binary files for demos.
- **etc**: Directory that contains other executables, such as daemons, microcode, and
  files relating to the license manager, for use primarily by system
  administrators.
- **examples**: Directory that contains sample source files of programs in MPL and MasPar
  Fortran, for reference by users learning to program in these languages.
- **field**: Directory that contains diagnostics routines, restricted primarily for use by
  MasPar Customer Service engineers. This directory is accessible only to the
  maspar and root logins.
- **include**: Directory that contains header files used in MPL (both ANSI-based and K&R
  C-based) and MasPar Fortran compilations.
- **lib**: Directory that contains MasPar libraries used in MPL (both ANSI-based and
  K&R C-based) and MasPar Fortran compilations.
- **log**: Directory that contains system log files.
- **.id**: File that contains the version number of the current MasPar software release.
In addition to the directories and files under `/usr/maspar`, the following directories are distributed with the MasPar software:

`/usr/man` Contains online man pages describing standard UNIX routines and MasPar-specific routines.

`/sys` Contains the modules necessary for rebuilding the kernel.

In these areas, the MasPar files are merged with the standard ULTRIX files.

**CAUTIONS:**

Do not move any of the MasPar directories or their contents. Also, it is recommended that you do not add things to these directories unless directed to do so by MasPar.

---

**Maintaining Multiple MasPar Directory Trees**

**NOTE:** The following information is only pertinent if you need to maintain multiple versions of the MasPar software.

When you build the ULTRIX kernel using `doconfig`, you will be linking the MasPar-provided object modules (in `/usr/sys`) associated with the most recently installed MasPar software release.

The ACU kernel, dpumanager, and DPU microcode running on the system must always be from the same MasPar software release as the one that generated the ULTRIX kernel. You should make sure that the MP_PATH environment variable in `/etc/rc.maspar` is set to the MasPar software directory that corresponds to the software version used to generate the ULTRIX kernel (in the case of this release, `/usr/kits/maspar/mp3.0_ds`). You should always set your MP_PATH to that directory when performing system administration tasks, such as starting or restarting the job manager.

---

**User maspar and Group maspar**

The MasPar system software installation process creates a user login account called "maspar" and a user group also called "maspar". The user ID is 109 and the group ID is 17. The software installed under the `/usr/kits/maspar` or `/usr/maspar` directory is owned by the user maspar and the group maspar.¹ When the software installation process creates a maspar login account, it does not assign a password. You need to assign a password to this account as soon as the installation is complete.

The user maspar login is intended for administrative use. Generally, MasPar Customer Support engineers will log in using this account. Members of the group maspar have privileges similar to

---

¹Except for some diagnostics routines, which are owned by "root".
the user maspar. Generally, all people who administer the MP-1 system should be members of this group. Group membership allows you to modify directories under /usr/maspar, start up the dpumanager process, run mptimelimit, and run MP-1 diagnostics.

**NOTE:** Membership in group maspar is not necessary for using the MP-1. MasPar Computer Corporation does not recommend assigning ordinary users to this group.
Chapter 2

Installing and Re-installing Software

When you purchase a MasPar® machine, the system software is already installed. The Installation Guide distributed with the MasPar system (and any new upgrades) describes how to install the software. This chapter discusses both installations from scratch and installations of upgrades; the information is a duplication of what is in the installation guide, but is presented again here to make it easier for you to find the installation information that you may need.

When you purchase an additional product from MasPar (such as MasPar Fortran or the MPIPL and MPML libraries), the Release Notes distributed with the product describes how you install and access it.

MasPar ships each MP-1 with the operating system and all the standard MasPar software already installed. Before beginning any installation procedures, do the following tasks:

1. Initialize and configure all devices that attach to the DPU before configuring the MP-1 system. Some devices, such as the MasPar Parallel Disk Array (MPDA), must be powered up before you power up the DPU (see Chapter 6). Consult your device manuals for additional details.

2. Read the Release Notes and Installation Guide for details about the software release, then follow the correct installation steps for your site and release.

3. Install any optional products according to the instructions in their release notes, and modify your license.dat file as instructed by MasPar Customer Support. (The licensing information is distributed to you with the documentation for the product. Refer to Chapter 5 for details on the license manager.)

4. Turn to the Installation Guide and follow the steps for verifying the installation.

5. If the installation was for an upgrade, place the new documentation (including the Release Notes and Installation Guide) in the binders, behind the tabs provided.

Once the installation is completed you need to perform the setup and configuration tasks detailed in the rest of this manual:

- configure the machine for a network or stand-alone environment
- set up the modem and configure the software
- set up Motif as the window manager
- set up the user’s environment variables
- set the dpumanager defaults for your system

**Installing ULTRIX 4.2a Software from Scratch**

As you work through the following steps, you will need to refer to the documentation for your CDROM reader. You may also want to refer to some of your DEC documentation for additional information.

**NOTE:** These installation instructions assume that your CDROM reader is installed as SCSI device number 4 (\$ 4) attached to the DECstation 5000. If it is installed as another number, substitute that number where you see \$ 4 in these instructions.

To install ULTRIX 4.2a:

1. Power the system on.

2. Confirm that your CDROM reader is SCSI device number 4 by typing:

   ```
   >> init
   >> cnfg 5
   ```

   The system responds with information on all attached devices. If the CDROM reader isn’t shown as \$ 4 you’ll need to adjust the reader; follow the directions in the reader’s documentation.
3. Follow the directions in the reader's documentation to place the MasPar System Software CDROM in the CDROM housing, and insert the MasPar CDROM into the reader. Different CDROM reader models have different methods for inserting the CDROM into the housing and the reader; be sure to check the instructions in the CDROM reader's documentation.

4. Type the following to boot up the system from the CDROM:

   >> boot 5/zz4/vmunix

This loads the kernel into memory and sets up a RAM-based file system.

5. The system displays a number of messages as it boots, then it displays a menu:

   1) BASIC Installation
   2) ADVANCED Installation
   3) System management

Since you will need to change a few of the default installation settings, choose 2) **ADVANCED Installation**, from this menu.

6. The system asks you to select which disk drive you wish to use as the system disk. Currently there is only one choice, drive zero (zz0).

   Select 1 for zz0, as the system drive.

   The system asks for confirmation; type y for yes.

   The system now creates the root file system, loads the root image, and halts. This takes about 9 minutes.

7. Boot the system by typing:

   >> setenv boot "5/zz0/vmunix -a"
   >> boot

8. The system asks a series of questions about the system name, the current date and time, your geographic region, and the system password. The system name is the host name for this machine, which should be entered in lowercase letters. You'll use the system name and root password again in later steps.

9. The system asks if you want to use the default file system layout. Type y for yes.

   The system proceeds to create the new filesystem; this takes a few minutes.

10. The system asks what kind of monitor is attached to the machine. Select the option appropriate to your system.
11. You now see a list of software subsets. The first items on the list are the subsets that must be installed, and that the system will install automatically. These items are unnumbered.

The second part of the list shows optional subsets that you can choose to install. The second part of the list contains both MasPar-required subsets and MasPar-recommended subsets. You must choose the following required subsets for the MasPar software to install and run correctly.

**Required optional subsets:**
- Doc. Preparation for Ref. Pages
- Software Development Utilities
- Internationalization Runtime Env.
- Communications Utilities

In addition, you should choose the following recommended subsets from this list:

**Recommended optional subsets:**
- Printer Support Environment
- Unix-to-Unix Copy Facility
- Ref. Pages for Sys. Admin. & Users
- Reference Pages for Programmers
- Worksystm Development Environment
- UWS Runtime Reference Pages
- UWS Development Reference Pages

The items in the second part of the list are numbered. Select the subsets you want to install by typing the menu number corresponding to the subset description on the command line. Enter all your entries on the same line, separated by spaces. When you are sure you have entered all the desired subsets, press Return.

The system responds with a list of all the subsets that will be installed (mandatory and optional) and asks for your confirmation. Check that the correct subsets are on the list. If not, type n (for no) and repeat the step. If the list is correct, type y.

The system now loads the selected software subsets. If you have selected the recommended subsets, this takes about 40 minutes.

**Note:** You can install other optional subsets later using the `set ld` command.

12. Once the base software subsets are installed, the system presents a menu of “mandatory upgrade” subsets that will be installed, followed by a list of “optional” subsets. **Do not choose any of the “optional” subsets on the optional subset list (choose None of the Above).**

Like in step 11, the system responds with a list of all the subsets that will be installed; in this case only the mandatory subsets should be listed. Check the list, and confirm that it is correct by typing y.

Allow about 20 minutes for the system to load the mandatory upgrade subsets.

13. The system next presents you with a list of kernel options. Choose None of the Above. The system asks for confirmation; type y.

14. The system asks if you wish to edit the kernel configuration file. Answer n for no.

The system now builds the ULTRIX kernel, creates the system device files, and reboots. After about 10 minutes you are presented with a login screen.
15. Log in as user `root`.

16. Create and mount the `/usr/kits` disk partition by typing:

   `# newfs -n /dev/zz0h rz56`

   The system responds with some messages about the partition. When the prompt
   returns, type the following.

   `# mkdir -p /usr/kits`
   `# cd /etc`

   Edit the file `fstab` (using whatever editor you prefer) and add the following line:

   `/dev/zz0h:/usr/kits:rw:1:3:ufs:`

   Save the changes to `fstab` and exit the editor. Then type:

   `# mount /usr/kits`

   The basic ULTRIX software installation is now complete.

---

**Installing the ULTRIX Software Upgrade**

Follow these set of steps if you are upgrading from any previous MasPar Software releases. The
following steps install the DEC 4.2a software upgrade.

1. Make sure the entire MasPar system is powered up, and you are logged in as user `root`.

2. Back up your `license.dat` file:

   `# cp /u0/maspar/*ds/etc/license.dat /usr/etc`

3. Follow the directions in the CDROM reader’s documentation to place the MasPar
   System Software CDROM in the CDROM housing, and insert the MasPar CDROM
   into the reader. Different CDROM reader models have different methods for
   inserting the CDROM into the housing and the reader, be sure to check the
   instructions in the CDROM reader’s documentation.

4. Complete the following steps to enter single-user mode and to mount the CDROM.
   Type:

   `# shutdown now`

   The system responds with some messages. When the prompt returns, type the
   following:

   `# /etc/mount -r /dev/zz4c /mnt`

   If you get an error, such as `Can't find device rz4c`, or the lights on the
   CDROM reader don’t go on, the CDROM reader isn’t attached correctly. Power
   down the MasPar system, then do steps 1, 2, and 3 on page 2-2.

5. You’ll use the `setld` command to load the MasPar software. Make sure you have
   enough space in these areas before beginning the load. If there is not enough space
   available, the system will fill the disk as much as possible, and then abort the
   installation. To see if you have enough space, type:
# df -l

setld needs about 30 MB in the /usr partition and about 3 MB in the root partition in order to load the software. Quick Tip: If you need space in /usr, and have installed Motif, you can delete Motif at this time since it is also contained on this CD. For instructions on how to reload Motif, see step 5 on page 2-7.

Load the MasPar software by typing:

    # /etc/setld -l /mnt/RISC/BASE_UPGRADE

Note: If you see the message setld: All subsets on the kit are already installed, your system is already upgraded to 4.2a, and step 5 is not applicable for your system. Proceed to the installation instructions for the MasPar System software.

6. The system presents a list of software subsets to install. The first part of the list contains subsets that must be installed; the system will install these unnumbered items automatically if they are not already present. The second part of the list contains optional subsets that you must choose to install. Since this is an upgrade, select None. Press Return to begin the load process.

Confirm your selections by typing y.

setld now loads the 4.2a upgrade software. This takes about 20 minutes. If setld is unable to load all of the subsets due to lack of space, installation will be aborted.

You can repeat steps 4, 5, and 6 to load additional subsets.

The DEC ULTRIX software upgrade is now installed. Your screen now displays several messages, including the instructions to run doconfig. Do not run doconfig at this time. If you installed the TrueColor subset, you may get a message indicating a mismatch with the kernel. The correct response to this message is provided in the following set of steps on upgrading the MasPar software.

## Installing V3.0 MasPar System Software

Follow these steps if you’re installing MasPar System Software for the first time.

1. To mount the CDROM, type:

    # /etc/mount -r /dev/zz4c /mnt

2. You’ll use the setld command to load the MasPar software. Make sure you have enough space in these areas before beginning the load (on a new installation this should not be a problem).

    setld needs about 38 MB in the /usr partition, about 99 MB in the /usr/kist partition, and about 3 MB in the root partition in order to load the software.

Load the MasPar software by typing:

    # /etc/setld -l /mnt/RISC/MASPAR

3. The system presents a list of software subsets to install. The first part of the list contains subsets that must be installed; the system will install these unnumbered
items automatically. The second part of the list contains optional subsets that you must choose to install. Following are the subsets contained in the MasPar V3.0 system software release:

**Mandatory subsets:**

ULTRIX V4.2a Base Software Update

**Additional subsets**

ULTRIX 4.2a Kernel Update  
MasPar Runtime System  
MasPar Field Service Utilities  
MasPar Programming Language  
MasPar Programming Examples  
MasPar License Manager  
MasPar Disk Array Utilities  
MPPE for DECstations  
MasPar Demonstrations  
MPPE for SUN/SPARC Workstations  
MasPar Fortran  
MasPar Image Processing Lib  
MasPar Mathematics Lib  
MasPar VAST II

**NOTE:**

If presented, you must select the License Manager subset from the optional list.

Type the menu number of your selections on the command line. Enter all your entries on the same line, separated by spaces. If you have purchased MasPar Fortran, VAST, MPIPL, or MPML, select those optional subsets also. Press Return to begin the load process.

Confirm your selections by typing y.

`setld` now loads the MasPar software. This takes about 45 minutes. If `setld` is unable to load all of the subsets due to lack of space, installation will be aborted.

You can repeat steps 2.3, 2, and 2.3 to load additional subsets.

4. OSF Motif is distributed as the recommended window manager. All MasPar software, including MPPE, will operate correctly without Motif installed; using Motif will give a consistent look and feel to MPPE and other MasPar products. Motif requires 18 MB of disk space. To see if you have enough space, type:

```
# df /usr
```

To load Motif, type the following:

```
# /etc/setld -l /mnt/RISC/MOTIF
```

The system presents a number of subsets to install. MasPar recommends you select ALL to install all of the subsets. When the system asks for confirmation of your selections, type y if they are correct.
The system takes approximately 10 minutes to load the Motif subsets.

5. If you will use the apropos or whatis ULTRIX commands, you should complete this step. If not, skip to the next step. (You use the apropos and whatis commands to get more information on man pages.) To make sure these two commands work correctly on your system, type:

```
# catman -w
```

Depending on your system configuration, this can take anywhere from 4 minutes to 20 minutes.

6. Before building the MasPar version of the operating system, remove the CDROM from the reader and install genvmunix as the current operating system by typing:

```
# /etc/umount /mnt
# cd /
# mv vmunix vmunix.old
# ln genvmunix vmunix
# /etc/shutdown -h now
```

7. Your DPU hardware should be connected, configured, and powered on. If it is not, follow these steps to set it up before proceeding to step 8.
   a. Power off the DECstation 5000.
   b. Connect the DPU.
   c. Power on the DPU.
   d. Power on the DECstation 5000.

8. Boot the system into single-user mode by typing:

```
>> boot 5/rz0/vmunix
```

The system boots into single-user mode. Verify from the startup messages on the screen that all the appropriate MasPar hardware is present and recognized by the operating system. In particular you should see a line starting with:

```
acu0 at vba0 ...
```

If you have a disk array attached to your system you should see:

```
ms0 at vba0 ...
```

For each disk array bank, there should be an entry like:

```
da0 at ms0 ...
```

If you have a Parallel VME board installed, you should see a line starting with:

```
iq0 at vba0 ...
```

If you do not see the appropriate lines for your hardware, stop here and do not proceed. Halt the workstation and make sure the DPU system is fully powered up and all the cables are connected. Contact MasPar Customer Support at 1-800-526-0916 for additional assistance. Once the problem is corrected, repeat this step and check the screen output. Do not continue the installation until you know the operating system recognizes all the MasPar hardware devices you have installed.

9. Once you have confirmed that all the appropriate MasPar hardware is present and recognized, use the doconfig utility to create a new vmunix customized to the system. Type:
# fsck -p

The system should respond with messages about "unmounted cleanly." If so, type:

```
# /etc/mount -a -t ufs
```

The system will mount /usr and /usr/kits. Next, type:

```
# /etc/doconfig
```

Answer the questions presented by doconfig:

a. Enter the name of your system in lowercase letters; use the same system name you used in step 8 during the ULTRIX Installation (see page 2-3).

b. The system asks A system with that name already exists. Replace it?. Type y.

c. Enter the current date and time, and enter all the time zone information.

d. When the system asks you whether you want to edit the configuration file, answer n.

The doconfig utility takes about 20 minutes to run. You will get status messages periodically while it is running.

At the conclusion of doconfig a new operating system image is located in /sys/MIPS/hostname/vmunix where hostname is the system name you gave doconfig, echoed in uppercase letters.

10. If you have a disk array, edit the /etc/fstab file and on the lines for the disk array file system, change ufs to mpfs.

11. Install the new vmunix by typing:

```
# rm vmunix.old vmunix
# mv /sys/MIPS/hostname/vmunix vmunix
```

where hostname is the system name you gave doconfig, echoed in uppercase letters.

Note that you have removed the old vmunix. This is because there is usually insufficient room in the root partition for more than one. Do not remove genvmunix.

The MasPar system software is now installed.

---

**Upgrading to V3.0 MasPar Software**

Before installing the MasPar system software upgrade, you must first install the ULTRIX 4.2a software upgrade.

To load the V3.0 MasPar software:

1. To ensure that disk space is available for the MasPar components, type the following:

```
# mkdir -p /usr/kits
```
# cd /etc

Edit the file \texttt{fstab} (using whatever editor you prefer) and add the following line:

\texttt{/dev/rr0h:/usr/kits:rw:1:3:ufs:}

Save the changes to \texttt{fstab} and exit the editor. Then type:

\texttt{
# /etc/mount -a -t ufs
}

2. You now need to remove any previous versions of the MasPar software. If you have material in the \texttt{/usr/kits/maspar} directory that you want to save, you need to be move it elsewhere or it will be deleted. Then type:

\texttt{
# rm -rf /usr/kits/maspar
}

You also may want to clean up some of the old control files for subsets installed in previous releases. If so, type:

\texttt{
# rm -f /usr/etc/subsets/MPD*
}

3. You use the \texttt{setld} command to load the MasPar software. Make sure you have enough space in these areas before beginning the load.

\begin{center}
If there is not enough space available, the system will fill the disk as much as possible, and then abort the installation.
\end{center}

To see if you have enough space, type:

\texttt{
# df -l
}

\texttt{setld} needs about 38 MB in the \texttt{/usr} partition, 99 MB in \texttt{/usr/kits}, and about 3 MB in the \texttt{root} partition in order to load the software.

Load the MasPar software by typing:

\texttt{
# /etc/setld -l /mnt/RISC/MASPAR
}

4. The system presents a list of software subsets to install. The first part of the list contains subsets that must be installed; the system will install these unnumbered items automatically. The second part of the list contains optional subsets that you must choose to install. Following are the subsets contained in the MasPar V3.0 system software release:
Mandatory subsets:

ULTRIX V4.2a Base Software Update

Additional subsets

ULTRIX 4.2a Kernel Update
MasPar Runtime System
MasPar Field Service Utilities
MasPar Programming Language
MasPar Programming Examples
MasPar Disk Array Utilities
MasPar License Manager
MPPE for DECstations
MPPE for SUN/SPARC Workstations
MasPar Fortran
MasPar Mathematics Lib
MasPar Image Processing Lib
MasPar VAST II
MasPar Demonstrations

NOTE:
If presented, you must select the License Manager subset from the optional list.

Type the menu number of your selections on the command line. Enter all your entries on the same line, separated by spaces. If you have purchased MasPar Fortran, VAST, MPIPL, or MPML, select those optional subsets also. Press Return to begin the load process.

Confirm your selections by typing y.

setld now loads the MasPar software. This takes about 45 minutes. If setld is unable to load all of the subsets due to lack of space, installation will be aborted.

You can repeat steps 3 and 2.4 to load additional subsets.

5. OSF Motif is distributed as the recommended window manager. All MasPar software, including MPPE, will operate correctly without Motif installed; using Motif will give a consistent look and feel to MPPE and other MasPar products. Motif requires 18 MB of disk space. To see if you have enough space, type:

```
# df /usr
```

To load Motif, type the following:

```
# /etc/setld -l /mnt/RISC/MOTIF
```

The system presents a number of subsets to install. MasPar recommends you select All to install all of the subsets. When the system asks for confirmation of your selections, type y if they are correct.

The system takes approximately 10 minutes to load the Motif subsets.

6. If you will use the apropos or whatis ULTRIX commands, you should complete this step. If not, skip to the next step. (You use the apropos and whatis commands to get more information on man pages.) To make sure these two commands work correctly on your system, type:
# catman -w

Depending on your system configuration, this can take anywhere from 4 minutes to 20 minutes.

7. Before building the MasPar version of the operating system, remove the CDROM from the reader and install genvmunix as the current operating system by typing:

```
# /etc/umount /mnt
# cd /
# mv vmunix vmunix.old
# ln genvmunix vmunix
# /etc/shutdown -h now
```

8. Your DPU hardware should be connected, configured, and powered on. If it is not, follow these steps to set it up before proceeding to step 9.
   a. Power off the DECstation 5000.
   b. Connect the DPU.
   c. Power on the DPU.
   d. Power on the DECstation 5000.

9. Boot the system into single-user mode by typing:

```
>> boot 5/zz0/vmunix
```

The system boots into single-user mode. Verify from the startup messages on the screen that all the appropriate MasPar hardware is present and recognized by the operating system. In particular you should see a line starting with:

```
acu0 at vba0 ...
```

If you have a disk array attached to your system you should see:

```
ms0 at vba0 ...
```

For each disk array bank, there should be an entry like:

```
da0 at ms0 ...
```

If you have a Parallel VME board installed, you should see a line starting with:

```
iq0 at vba0 ...
```

10. Once you have confirmed that all the appropriate MasPar hardware is present and recognized, use the `doconfig` utility to create a new `vmunix` customized to the system. Type:

```
# fsck -p
```

The system should respond with messages about "unmounted cleanly." If so, type:

```
# /etc/mount -a -t ufs
```

The system will mount `/usr` and `/usr/kits`. Next, type:
# /etc/doconfig

Answer the questions presented by doconfig:

a. Enter the name of your system in lowercase letters; use the same system name you used in step 8 during the ULTRIX Installation (see page 2-3).

b. The system asks A system with that name already exists. Replace it?. Type y.

c. Enter the current date and time, and enter all the time zone information.

d. When the system asks you whether you want to edit the configuration file, answer n.

The doconfig utility takes about 20 minutes to run. You will get status messages periodically while it is running.

At the conclusion of doconfig a new operating system image is located in /sys/MIPS/HOSTNAME/vmunix where HOSTNAME is the system name you gave doconfig, echoed in uppercase letters.

11. If you have a disk array, edit the /etc/fstab file and on the lines for the disk array file system, change ufs to mpfs.

12. Install the new vmunix by typing:

    # rm vmunix.old vmunix
    # mv /sys/MIPS/HOSTNAME/vmunix vmunix

where HOSTNAME is the system name you gave doconfig, echoed in uppercase letters.

Note that you have removed the old vmunix. This is because there is usually insufficient room in the root partition for more than one. Do not remove genvmunix.

13. You now need to make sure that your server matches your console type. Type:

    # /etc/server_scps

You'll see a numbered selection for color, black and white, and greyscale monitors. Select the one that's appropriate for your system.

14. Next, restore your license.dat file by copying it back from its temporary storage place:

    # cp /usr/etc/license.dat /usr/maspar/etc
Installing and Re-installing Software
Chapter 3

Configuring Your Installation

This chapter discusses tasks you need to perform to configure and tune the MasPar software so it works properly in your environment:

- configure your system as either a stand-alone machine or else to run in a network environment
- set up the internal modem
- set up Motif as the window manager
- set environment variables
- add new users to the system
- set up the system to run (or display) on a remote workstation
- set up Yellow Pages
Configuring For a Network Environment

**CAUTIONS:**

Make sure the system is shut down as much as possible before you run `netsetup`.

Do not run `netsetup` while you have any user jobs running.

To configure your system to work as part of a local area network:

1. Bring the system into single user mode by typing `shutdown now`.
2. Type `netsetup`.
3. When you finish running `netsetup`, reboot the system.
4. Make sure the system host name is registered in the `/etc/hosts` file.
   In a network environment, the `realhostname` name must appear on a separate line. The host name must not be on the `localhost` line.
5. Verify that the `/etc/rc.local` file sets the host name. Near the beginning of the `/etc/rc.local` file, check for a line like:

   `/bin/hostname realhostname`

   This `realhostname` must be the same as the one in `/etc/hosts` and `/usr/maspar/etc/license.dat`.

For more information, see your ULTRIX documentation regarding setting up networks.

Configuring For a Stand-Alone Environment

To set up your MasPar machine to work in a stand-alone environment, add the host name of your MasPar machine at the end of the `localhost` line in `/etc/hosts`, as follows:

```
127.0.0.1 localhost realhostname
```

This aliases the MasPar machine to `localhost`. Make sure `realhostname` is not present anywhere else in the `/etc/hosts` file.
Setting Up and Using the Modem

**CAUTIONS:**

These instructions are for systems sold in the U.S., which are equipped with an internal modem. Outside the United States, all systems use an external modem and different configuration instructions. If your system has an external modem, use the instructions supplied with the modem.

The MasPar modem circuit requires a line using tones (not dial pulses).

Configuring Software for the Modem

To configure the MasPar internal modem's software:

1. Set the modem switch on the DPU front inner door to CONNECT.

2. Log in as root.

3. Run the setup script to set up the modem configuration files. The setup script asks you questions that identify the type of modem attached to the system and what it is attached to. Start the setup script by typing:

   ```bash
   # uucpsetup -m
   ```

   The script asks for the following information; answer by typing the items in boldface below each question.

   How many modems are you adding to this system [1]?

   1

   Enter the MODEM TYPE of modem number 1, "?" for help:

   `tbfast`

   Enter the TTY LINE to attach modem number 1 to, "?" for help:

   The number you type for the TTY LINE depends on the port the cable is connected to. If you're connected to Serial Port 2 (as numbered on back of the machine), type:

   ```bash
   tty00
   ```

   Enter the SPEED (baud rate) modem 1 runs at, "?" for help. [9600]:

   9600

   Enter "out", "in", or "shared" [shared]:

   `shared`

4. Edit the `/etc/mtys` file. Replace the line starting with `tty00` with:

   ```bash
   ttyd0 " /etc/getty std.9600" dialup off modem shared
   ```

5. At the system prompt, type:

   ```bash
   kill -HUP 1
   ```

6. At the system prompt, type:

   ```bash
   tip 0.9600
   ```

   This opens `tip` and provides a local connection to the modem.
7. At the system prompt, type the following:
   \texttt{AT \$S63=1}
   \texttt{AT \&F S7=60 S45=255 S51=255 S52=002 S54=003 S58=002 S64=1 S66=1}
   \texttt{AT S94=0 S95=2 S110=001 S111=30 S130=5 S131=1 Q6 E0 \&W0}
8. End the \texttt{tip} session by typing a tilde and a period:
   \texttt{\~.}
9. Edit the \texttt{/etc/ttys} file again. On the \texttt{ttyd0} line, replace \texttt{off} with \texttt{on}.
10. At the system prompt, type:
    \texttt{kill \_HUP 1}

\begin{center}
\textbf{CAUTION:}
\end{center}

Switching the modem switch between \texttt{DISCONNECT} and \texttt{CONNECT} can sometimes cause the DECstation 5000 to freeze up. If this happens, a reset is required to restore the machine.

\section*{Using the Modem}

To use the modem, follow these steps:

1. Check to see that the modem switch on the DPU front inner door is set to \texttt{CONNECT}.

2. At the system prompt, type:
   \texttt{tip d0.9600}

3. At the system prompt, type:
   \texttt{ATE1 ATDTphonenumber}

   If you have reached a modem, you will be connected to it. If you have not, the
   modem will time out, and you will have to redial the number using the procedure
   just described.

4. End the \texttt{tip} session when you are done with the modem. At the system prompt,
   type a tilde and a period:
   \texttt{\~.}

\section*{Setting Up the Motif Window Manager}

OSF Motif is distributed as the recommended window manager. You may have made Motif the default window manager during the installation (see step 5 of "Installing V3.0 MasPar System Software" on page 2-7). All MasPar software, including MPPE, will operate correctly without Motif installed; however, using Motif will give a consistent look and feel to MPPE and other MasPar products.
If you didn’t install Motif during the software installation, and now wish to install it, you first need to check and see if you have enough disk space; Motif requires 18 MB of disk space. To see if you have enough space, type:

```
# df /usr
```

To load Motif, insert the MasPar CDROM in the CDROM reader and type:

```
# /etc/setld -l /mnt/RISC/MOTIF
```

The system presents a number of subsets to install. MasPar recommends you select **All** to install all of the subsets. When the system asks for confirmation of your selections, type **y** if they are correct.

The system takes approximately 10 minutes to load the Motif subsets.

To see if you’ve installed Motif correctly, type:

```
# setld -i | grep -e DXM
```

the system should respond with a non-zero number. (If the number is less than 10, it may be that not all of the Motif subsets were installed; you may want to reinstall Motif if this is true.)

To make Motif the default window manager on your DECstation:

1. Select Window from the Customize menu in the DECwindows Session Manager.
2. On the Window Manager line, enter the pathname for Motif:
```
/usr/lib/DXM/clients/mwm/mwm
```
3. Log out and log back in. Motif will now be your default window manager.

If your users will be using a DECstation other than the MasPar front end to display MPPE (by displaying MPPE remotely on that workstation), then you should also make Motif the default on their DECstations.

**Setting Up Your Environment**

Certain environment variables must be set for your MasPar software to operate properly. These include the DISPLAY variable and several variables particular to MPPE: MP_PATH, MP_DBTARGET, and MP_NODEBUG. This section only deals with MP_PATH; the other variables are discussed in Appendix A of the *MPPE User Guide*.

All MasPar software uses the MP_PATH environment variable. MP_PATH points to the top of the MasPar directory tree on your system. (See page 1-2 for a discussion of the directories and files in the MasPar directory tree.)

Use the `setenv` command to set the MP_PATH. Type the following lines at the system prompt to make this effective for the current session only. To make it effective permanently, add these lines to your `.cshrc` file.
setenv MP_PATH /usr/maspar
set path=($MP_PATH/bin $MP_PATH/etc $MP_PATH/field/bin $path)

In this example, MP_PATH describes the variable, and $MP_PATH describes the value that the variable is set to.

By default, if no MP_PATH is specified, MasPar programs use /usr/maspar. /usr/maspar is linked to /usr/kits/maspar/mpx.x_ds, where xx is the MasPar release number. This link is automatically created by the installation and update scripts.

If you installed the MasPar software in a directory different from the default, change /usr/maspar (using a symbolic link) to be the top of the tree where you put the maspar software. Alternatively, you can create a link on the MasPar machine called /usr/kits/mp_ds. Then, link /usr/maspar on the remote machine to /usr/kits/mp_ds; then when you update the MasPar software you only have to update the one link on the MasPar machine.

NOTE: If your site has multiple MasPar machines (or if you have previous releases still installed) you need to make sure that the MP_PATH in all user’s .cshrc files and MPPE’s MP_PATH setting in the Connect window are set to the same directory. When a user invokes MPPE, it will default to the MP_PATH specified in the .cshrc file, but the user may not realize that two settings are different.

Adding New Users

When you add a user to your MasPar system, make sure you do the following:

1. Define the environment variable MP_PATH in the user’s .cshrc file. By default MP_PATH is set to /usr/maspar. MasPar software, if run with MP_PATH undefined, assumes that /usr/maspar heads the MasPar directory tree. If this isn’t true for your system, in /usr on the user’s workstation link maspar to the “real” MasPar directory tree.

2. Include $MP_PATH/bin, in the user’s path (see .template and .cshrc on the MasPar workstation disk). It is a good practice to make this conditional:

   if ( ! $MP_PATH )
   then setenv MP_PATH /usr/maspar
      set path = ($MP_PATH/bin $MP_PATH/etc $MP_PATH/field/bin)
   endif

3. Make sure users who use rlogin to access MPPE on the MasPar workstation include a setenv DISPLAY line in their .cshrc files on the remote machine.

4. Include MasPar software mounts in /etc/fstab on the user’s workstation.

5. Make sure the MP_PATH directory tree is accessible (via NFS) to all LAN users.

6. Make sure /etc/svc.conf on the MP-1 system is set up correctly.

7. Make sure appropriate user accounts exist on the MP-1 DECstation, with rlogin or rsh allowed, since users running executable programs involving the DPU must do so on the MP-1 DECstation.
Remote Setup

This section tells you how to set up the system so that users can run MasPar software remotely. While the user’s program normally must be invoked on the DECstation that is directly connected to the MasPar DPU, MPPE can be invoked on a remote DECstation or on a SPARCstation, connected via Ethernet or on a LAN. In addition, the compiler drivers and other MasPar tools that do not use the DPU can be invoked on any DECstation that has access to the MasPar software on the MP-1 DECstation.

Running on a Remote DECstation

NOTE:
Make sure all remote DECstations on which you want to use the MasPar software are running ULTRIX V4.2a. You can determine whether a machine is running ULTRIX V4.2a by looking at the file /etc/motd (or by running `uname -a`).

To run the MasPar software on a remote DECstation, complete the following steps:

1. The MP-1 DECstation host name must be visible to the remote DECstation. Verify this by entering the following command on the remote DECstation:

   ```bash
   % /etc/ping MP1DECstationName
   ```
   If the host name is unknown, add it.

2. Each remote DECstation host name must be visible to the MP-1 DECstation. Verify this by entering the following command on the MP-1 DECstation 5000:

   ```bash
   % /etc/ping remoteDECstationName
   ```
   If the host name is unknown, add it.

3. Use `nfssetup` to export `/usr/kits/maspar/mpx.x_ds` from the MP-1 DECstation 5000 to remote DECstations, where “x.x” is the MasPar software release number.

4. Make a directory called `/usr/kits/maspar/mpx.x_ds` on the remote DECstation.

5. NFS mount `/usr/kits/maspar/mpx.x_ds` onto the directory you created in the previous step.

6. On remote DECstations, execute the command:

   ```bash
   % ln -s /usr/kits/maspar/mpx.x_ds /usr/maspar
   ```

7. The services ‘‘maspard’’ and ‘‘mdinfo’’ must be visible to the remote DECstation. If you are running Yellow Pages, you can verify this by entering the following command on the remote DECstation:

   ```bash
   % ypcat services | grep mdinfo
   ```
   If you are not running Yellow Pages, the following two lines must be in the file `/etc/services`:

   ```
   maspard 1653/tcp
   ```
mdinfo 1654/udp

8. All directories in which users will debug programs must be “visible to” (NFS mounted onto) the MP-1 DECstation 5000 and the remote DECstation—preferably mounted as the same name on both machines. If the mount points are different, individual users of MPPE will have to change the Directory setting in the Connect window. See the MPPE User Guide for more information on the Connect window.

9. Make sure each user’s user id (uid) is the same on each remote machine as it is on the MP-1. If it is different, the user needs to change the Login setting in the Connect window in MPPE. Otherwise, MPPE will fail with a permission problem when the user tries to execute a program within MPPE.

Running on a SPARCstation

NOTE:
Make sure all SPARCstations on which you want to use the MasPar software are running SunOS V4.1 or later. You can determine whether a machine is running SunOS V4.1 by looking at the file /etc/motd (or by running uname -a).

The only MasPar product that runs on a SPARCstation is MPPE. To run MPPE on a SPARCstation, complete the following steps:

1. The MP-1 DECstation host name must be visible to the SPARCstation. Verify this by entering the following command on the SPARCstation:

   `% /etc/ping MP1DECstationName`

   If the host name is unknown, add it.

2. Each SPARCstation host name must be visible to the MP-1 DECstation. Verify this by entering the following command on the MP-1 DECstation:

   `% /etc/ping SPARCstationName`

   If the host name is unknown, add it.

3. Use nfssetup to export /usr/kits/maspar/mpx.x_sun from the MP-1 DECstation 5000 to target SPARCstations, where “xx” is the MasPar software release number.

4. Make a directory called /usr/kits/maspar/mpx.x_sun on target SPARCstations.

5. NFS mount /usr/kits/maspar/mpx.x_sun onto the directory you created in the previous step.

6. On target SPARCstations, execute the command:

   `% ln -s /usr/kits/maspar/mpx.x_sun /usr/maspar`

7. The services “maspard” and “mdinfo” must be visible to the SPARCstation. If you are running Yellow Pages, you can verify this by entering the following command on the SPARCstation:
% ypcat services | grep mdinfo

If you are not running Yellow Pages, the following two lines must be in the file /etc/services:

    maspard 1653/tcp
    mdinfo 1654/udp

8. All directories in which users will debug programs must be visible to (NFS mounted onto) the MP-1 DECstation and the SPARCstation—preferably mounted as the same name on both machines. If the mount points are different, individual users of MPPE will have to change the Directory setting in the Connect window in MPPE. See the MPPE User Guide for more information. For example, if you want to run under MPPE some of the programs in the MP-1 DECstation 5000 /usr/maspar/examples directory, you need to NFS mount /usr/kits/maspar/mpx.x_ds on the SPARCstation (see steps 3 through 2.3 above).

9. Make sure each user’s user id (uid) is the same on each remote machine as it is on the MP-1. If it is different, the user needs to change the Login setting in the Connect window in MPPE. Otherwise, MPPE will fail with a permission problem when the user tries to execute a program within MPPE.

10. Make the license.dat file visible to MPPE software. You can use either one of the following two methods to do this:

    - On the MP-1 DECstation 5000, copy the file /usr/kits/maspar/mpx.x_ds/etc/license.dat to /usr/kits/maspar/mpx.x_sun/etc. Note that if you modify the license file, you must make the change in both license.dat files (the one in the "_ds" directory and the one in the "_sun" directory).

    - Remote mount the "_ds" file system onto the SPARCstation and link the license file:

      ```
      ln -s /usr/kits/maspar/mpx.x_ds/etc/license.dat /usr/kits/maspar/mpx.x_sun/etc/license.dat
      ```

**Updating a Host Name**

The system host name must be consistent throughout the system files. If the host name changes, you can update the host name in the system files by running /etc/netsetup and answering the questions presented.

**NOTES:** Enter the system host name in all lowercase letters.

If the host name does not appear in the /etc/hosts file, the maspard daemon will not run, and MPPE will fail. See hosts(5) in the ULTRIX man pages for more information on the format of the /etc/hosts file.

Changing the host name also affects the license.dat file. This file is used by the licensing facility to define the licensed features and the server where the licensing software resides. If you change the host name, you must edit the license.dat file:

1. Make a backup copy of your license.dat file.
2. Open your license.dat file in your text editor.
3. Go to the line that begins with SERVER.
4. The first field after the word SERVER is the host name. Replace the old name with the new. NOTE: Do not change the host ID field on that line.

5. Save your changes, and exit your editor.

6. Run lmdown, then lmstart to make the change take effect.

Refer to Chapter 5 for more information on the license manager and the license.dat file.

**Yellow Pages Setup**

If you have already been running MPPE remotely under earlier releases, you must make sure the new services maspard and mdinfo are visible to the remote DECstation. You also need to add your hosts to the YP master hosts file.

To verify if you are running Yellow Pages, enter the following command on the remote DECstation:

    % ypcat services | egrep 'mdinfo|maspard'

If the frontend DECstation is not the Yellow Pages master, copy the lines containing the string maspard and mdinfo from the file /etc/services on the DECstation to the /etc/services file of the Yellow Pages master.

If you are not running Yellow Pages, the following two lines must be in the file /etc/services:

```
maspard    1653/tcp
mdinfo     1654/tcp
```
Chapter 4

File System Management

This chapter explains how to set up and use the various file systems available for disk arrays and IORAM disks:

- which file systems can be NFS exported
- how to set up and use the VME file system
- how to set up and use the IORAM file system
NFS Exporting of MPFS and IORAM File Systems

With the 3.0 MasPar software release, the disk array is mounted as file system type mdfs. In addition, an IORAM disk may be mounted as file system type ioram. These file systems may be exported over NFS, with the following caveats:

- Any local file system that is to be NFS exported must be locally mounted before the NFS server daemon (nfsd) is started. By default, MasPar mounts MPFS and IORAM file systems in /etc/rc.maspar, which is executed after /etc/rc.local, after the NFS daemons are started. An administrator desiring to export an MPFS or IORAM file system should move the relevant mount commands into /etc/rc or /etc/rc.local, before the NFS daemons are started.

- MPFS file systems may be larger than 2 GB, which is the limit under the ULTRIX file system (UFS). Large MPFS file systems may be exported to NFS. We have observed that NFS client systems running ULTRIX or SUNOS can access the large file systems normally. However, the getmnt system call and the df command will return bogus values for the file system size and blocks free. This is caused by an integer overflow in the NFS code in the client workstation.

VMEFS file system

For the V3.0 release of the MasPar software, the "vmeaccess" functions which allow programs to directly access devices on the MasPar VME bus have been rewritten. Regions of the VME bus may now be viewed as ordinary UNIX files and the MasPar plural I/O functions (p_read, p_write, read2dcs, etc.) may be used to access those files.

To implement this MasPar has developed a new file system type, called vmefs, and incorporated it into ULTRIX. This file system is mounted just like a disk or NFS file system. Each file within this file system represents a region of the VME address space, with special attributes including a base VME address and the addressing mode (such as A32D32).

Use the mount command to initially mount the VME file system:

```
  mount -t vmefs <device> <mount point>
```

<device> is normally /dev/vme (the name is not actually used except as a label) and <mount point> is the name of an existing directory where this file system is to be mounted, such as /vme.

When first mounted, the VME file system is always empty. Use the mount and df commands to show the directory with the vmefs file system as mounted.

The vmeaccess(3) library functions have been recoded to use the vmefs file system. These functions presume that the vmefs file system is mounted as /vme. Most programmers will probably prefer to use vmeaccess(3) rather than the lower level primitives described here.

To create a VME file, use open or creat system calls to create a zero-length node with no associated VME address or mode. Then use the ioctl call VMEFSENABLE on the file to set the
base VME address, the address mode, the size of the region, and other attributes. When you are done, `ls` will show a file with the name you specified and with a size equal to the region size. This file will persist until it is unlinked or the file system is unmounted.

To see the full attributes of the VME file, use the `ioctl` call `VMFSSTAT` on the file. This returns the same information that was specified by `VMEFSENABLE`. To print the information in a readable form, type:

```
# vmestat <file list>
```

Once created, the VME file may be used just like a regular file, with the following restrictions:

- The file may not be extended in size except by use of the `ioctl(2)` call, `VMEFSENABLE`. However, it may be truncated to a shorter length in the normal manner. You will get an error if you try to open the file in append mode.

- The `read` and `write` system calls are not currently supported and return an error. Among other things, that means `cp` does not work. Only the plural I/O functions based on the new system calls, `mpbufget` and `mpbufput` (which include `p_read`, `p_write`, `pp_read` and `pp_write`), will work on VME files.

Control of byte swapping on plural I/O functions is the same as for other types of files. Briefly, the `O_BIGENDIAN` flag may be passed to `open` to specify that the file data is in big-endian form. Otherwise the data is assumed to be little-endian. Use `p_fcntl` to specify the size of the data (8, 16, 32 or 64 bits) if the data is big-endian.

The programmer may tell the file system to use PVME or IORAM buffers when reading or writing a VME file. This is specified with a flag to `VMEFSENABLE` and is consequently an attribute of the file. This flag has no effect if there is no PVME or IOCTLR in the system. Another flag tells the system to use block-mode DMA when transferring to/from the PVME or IOCTLR.

Unlike ordinary files, front-end memory mapping is permitted on VME files. This is accomplished by using the `ioctl` call, `VMEIOMAP`, as previously documented for `/dev/vme`. As before, the size of the memory mapped region is limited by availability of VME mapping registers (a total of about 3.5MB can be mapped in on a DSS000). For memory mapping only, the byte swap flags (MP_S8, MP_S16 or MP_S32) passed to the `ioctl` call, `VMEFSENABLE`, are used to control byte swapping; the `p_fcntl` call is not used.

The VME file system itself has certain restrictions which must be observed:

- File names are limited to 23 characters in length.

- There are only 64 inodes in the vmefs file system, including the root inode. Thus, less than 64 VME files can simultaneously exist even if unlinked. Note that `enableVme` holds an inode until the calling process exits.

- Subdirectories (other than `"."` and `".."`) are not supported. The `mkdir` and `rmdir` system calls are not supported by vmefs.

- Symbolic links may not be created within the vmefs directory. The `symlink` system call is not supported by vmefs. However, symbolic links outside vmefs may be followed into the vmefs directory. Hard links are fully supported.
The vmefs file system structures are maintained only in memory and are consequently volatile -- the directory is destroyed when the file system is unmounted or the machine is rebooted.

For compatibility with older diagnostic programs, the /dev/vme device driver interface has been maintained intact. Thus, programs that open /dev/vme and use it's memory mapping scheme should still work.

The IORAM File System

The IORAM file system is a local file system implemented under the Generic File System Interface, GFSI, which is described in the man page gfsi(5). This file system is memory resident, in MasPar IORAM or PVME memory. Access to an IORAM file is much faster than to any disk file (UFS or MPFS) since there is never any seek time or rotational latency. In addition, IORAM files can be directly accessed from the MasPar DPU using DPU I/O instructions, without first copying the data into accessible buffers.

An IORAM file system is generated as part of the mount operation. The mkfs and fsck commands are not used. A file system is normally mounted, using the mount command, as follows:

```
    mount -t ioram <device> <mount point>
```

Where -t ioram specifies the type of the file system, <device> is normally /dev/iq, and /ioram is the mount point (an existing directory). The newly mounted file system is always empty, with just a root directory. Permissions on the IORAM root directory are initially inherited from the mount point directory.

The following line must be present in the operating system configuration file for the IORAM file system code to be included:

```
pseudo-device ioramfs
```

This line must be added manually.

Space is allocated to the IORAM file system using rules specified in the file /usr/sys/data/ioram_config_data.c.

The operating system image, /vmunix, must be rebuilt after this file is modified or if the pseudo-device line is added to the configuration file. Refer to the ULTRIX System Manager manual, "Guide to System Configuration File Maintenance," for further information.

Except as noted here, the IORAM file system and IORAM files behave the same as the ULTRIX file system (UFS). The differences are:

- There is only one level of directory, except for "." and "..". The mkdir and rmdir system calls are not implemented in this file system.
- The root directory is limited to 64 entries, including "." and "..".
• File names are limited to 23 characters.

• There are a total of 64 inodes, including the root directory. This limits the total number of files, including unlinked but still referenced files, to under 64.

• There are a total of 62 indirect blocks. One indirect block is required for any file larger than 3MB. Three indirect blocks are required for files larger than 259MB. Each additional 256MB takes one more indirect block.

• Symbolic links cannot be created in this file system; the symlink system call is not supported. However, symbolic links may be used to point to an IORAM file.

• The file system is memory resident and therefore volatile. The contents of the file system are lost irrevocably when the file system is unmounted or the system reboots.

• Space is allocated to files in 256K blocks. There are no fragments. Thus the file system is not efficient for storage of small files.

• Commands that directly read the disk and UFS structures, including chpt, fsck, mkfs, dump, and restore will not work with this file system.

• The IORAM file system may be mounted multiple times at different mount points. This is a way of getting around the directory size limitations since there is a root directory for each mount point. However, IORAM memory comes from a common pool so this does not increase the space availability.

For more information, see the man pages for getmnt(2), getdirent entries(2), mount(2), mbufget(2), fstab(5), mpfs(5), and mount(8).
Chapter 5
Utilities and Daemons

This chapter describes the MasPar-specific utilities and daemons that aid system administration. This chapter discusses:

- the license manager, which controls access to various MasPar products
- the maspardi daemon, which manages communication for MPPE
- the DPU job manager, dpumanager, which controls the queue for the DPU through context switching (DPU access) and by swapping jobs to disk
- other MasPar utilities, including mpconfig, mpctl, mpfile, mpi, mplimit, mpq, mpstat, mpshutdown, mpswopt, mptimelimit, and mpzap
Working with the License Manager

The license manager controls the number of users that can use a MasPar program at a time. This software licensing facility consists of three visible files and a small number of administrative executables.

NOTE: If your installation has other license managers (or if you plan to acquire others later), beware of name conflicts (e.g., two files named license.dat). The safest way to avoid multiple license manager conflicts is to keep the MasPar license manager software and license.dat on the MasPar front-end workstation, and install other license managers on other workstations in the local network.

License Manager Files

The following files are part of the license-checking process and control the software licensing:

- /usr/maspar/etc/lmgrd—the license daemon
- /usr/maspar/etc/mplicensed—the MasPar license manager daemon
- /usr/maspar/etc/license.dat—the license file. This is an ASCII file with an encrypted string for verification; it contains the licensing data. If your license is updated, you need to change this file, as described later in this chapter in the section "Updating a license.dat File."

The license manager maintains a log of the states of the license manager in a log file called /usr/maspar/log/lmgrd.log.

license.dat

The license.dat file defines the server on which the licensing software resides, the licensing daemons residing on the server, and the licensed features.

Following is an example of a default license.dat file. It shows 1 MPPE core user (debug session), 4 simultaneous MPL users, and 16 users executing the MPPE graphic user interface (mppe-front).

SERVER maspar 08002b17b67c 1700  
DAEMON mplicensed /usr/maspar/etc/mplicensed  
FEATURE mppe mplicensed 99.000 1-Sep-99 1 98665C4343C3A9A78DF3 "MasPar License"  
FEATURE mpl mplicensed 99.000 1-Sep-99 4 DB962C7328333860D026 "MasPar License"  
FEATURE mppe-front mplicensed 99.000 1-Sep-99 16 8BB756E67026380C8B42 "MasPar License"

The "SERVER" line has three fields.

- The first field contains the host name as set in the /etc/rc.local file; here, maspar is the host name. You can edit this field.

- The second field contains the host ID; here, 08002b17b67c is the host ID. NOTE: Do not edit this field; doing so will invalidate the license file.

- The last field contains the host port number the license manager uses, as seen in the /etc/services file; here, 1700 is the host port number. You can change this field using your editor, or you can change it with the mpzap command. For details on this command, consult the man page.
Quick Tip: If you think you have a problem with the license manager, back up the license.dat file to a new name, then run lmdown to take the license manager down. Type mpcreat license to create a new license.dat file, then test to see if indeed it was the license.dat file causing the problems.

Updating a license.dat File

To change a license.dat file after you have received update information from MasPar, thus activating the updates, follow these steps:

1. Make a backup copy of your current license.dat file.
2. Enter the following command:
   
   ```
   # cd $MP_PATH/etc
   ```
3. Edit the license.dat file by following these steps:
   a. Remove all entries for the updated products in your license.dat file, but do not delete lines containing the names of products that you are not updating. For example, for MPPE updates, remove any lines with mppe in them.
   
   b. Add the license strings in the update information to your license.dat file.
   
   c. Make a backup copy of the new license.dat file.

   **NOTE:** Do not delete lines unless they are being replaced.
4. Enter these commands to restart the license manager daemon:
   
   ```
   # lmdown
   # lmstart
   ```
5. Test your new features.

Conflicts With ULTRIX

Some of the programs associated with the MasPar license manager, such as lmstat, were created on systems running older (pre-4.0) versions of ULTRIX. Consequently, these programs require the existence of an otherwise obsolete file, /etc/svorder, when doing hostname look up in a network environment. If this file does not exist on the local system, you are likely to get a message like:

```
> /usr/maspar/etc/lmstat
lmstat - Copyright (C) 1989, 1990, Highland Software, Inc.
Flexible License Manager status on Wednesday 7/31/91 17:32

License server status:

maspar: Cannot find host in network database

Vendor daemon status (on ):

mplicensed: bad SERVER hostname in license file
```

This problem only occurs when running license manager utilities on machines other than the MasPar system (for example, on a remote workstation).
Following is the contents of the DEC-supplied /etc/svorder file from a system running yellow pages:

```
# The /etc/svorder file designates the order and selection of
# ULTRIX name services to be queried in the resolution of host
# names and addresses. This file is not required for /etc/hosts
# (local) access, but is required to access host names from
# the current ULTRIX Yellow Pages and BIND services.
# Note that additional preparation is required to set up each
# name service. Consult the documentation for further information.

YP
local
```

The order of service names in the file specifies the lookup order. Valid names are "local", "yp", and "bind".

**Conflicting Licenses**

The MasPar software is controlled by Version 1.5 of the Highland license manager. If you have other software products that also use the Highland license manager, it might have problems recognizing your MasPar product licenses.

You can merge your license files provided the other licensed products also use Version 1.5 of the Highland license manager. In the merged license.dat file you should keep one of the "SERVER" lines, all of the "DAEMON" lines, and all of the "FEATURE" lines. Replace all the old license.dat files with this new one. Then restart the license manager process with "lmstart".

**Restoring a license.dat File**

If your license.dat file is destroyed, a default license.dat file is generated when the system boots. You can also create one manually by entering the following commands at the system prompt:

```
# cd $MP_PATH/etc
# ./mp_create_license > license.dat
```

If you have additional licenses, but do not have a backup license file, contact MasPar Customer support by email at support@maspar.com, or by phone at 1-800-526-0916, for a copy of your modified license.dat file.

**Quick Tip:** You may want to make several backup copies of your license.dat file. Then if you have a problem, you won't have to run with only the defaults (one MPPE session and 4 MPL sessions and nothing else) while waiting for a new license.dat file from MasPar.

**NOTE:** If you receive a new license.dat file by electronic mail, append the email to the old license.dat file. MasPar won't send the entire license.dat file by email, only the lines with new information.
When License Limits Are Exceeded

When any licensed product (such as MPL, MasPar Fortran, or MPPE) executes, it checks with the license manager software (lmgrd and mplicensed) for the number of users currently using the program. If the licensing limits are already met, the execution attempt fails, and the license manager tells the user that the system is not licensed for that many users. The language compilers and MPPE put you in a queue while waiting for a license to become free. To get out of this blocked state, enter <Ctrl>-C at the prompt, or in the MPPE Stdin/Stdout window.

Contact your MasPar Sales Representative if you would like to increase your license limits.

If you have purchased license upgrades to existing products, follow the instructions in "Updating a license.dat File."

License Manager Utilities

The following utility programs, which require root permissions to run, are supplied as standard with the license manager:

**lmdown**

Shuts down the license manager. Within approximately 5 minutes, all currently running user processes will die, even if the license manager is restarted. To continue, all user processes must be restarted.

**lmstart**

Starts (restarts) license manager services. Use it after an lmdown or at boot time. It is called from /etc/rc.maspar.

**lmhostid**

Displays the host ID of the currently running machine.

**lmlist**

Displays the users currently using a particular feature. For example:

```
lmlist mpf
```

Lists the current users of MasPar Fortran.

**lmremove**

Removes a single user’s license for a specified feature. This could be required in the case where the licensed user was running the software on a node that subsequently crashed. lmremove will allow the license to return to the pool of available licenses. For example,

```
lsremove [-c file] feature user host
```

lmremove removes all instances of user on node host from usage of feature. If the optional -c file is specified, the indicated file is used as the license file.

**lmstat**

Displays the status of the license manager software, indicating which servers and daemons are up.

**mp_create_license**

Prints to standard output the default license file for the current machine. It is referenced in the /etc/rc.maspar file. If there is no license file, the MP-1 uses mp_create_license to create one.
The maspard Daemon

The maspard daemon manages communication for MPPE (passing socket numbers, for instance), including AVS hookup. It also services mpi and mpq requests. It starts automatically on the DPU when booted (if in rc.local). It must be running on a machine for MPPE to connect to it.

Only one maspard process can run on the DPU or a particular machine. The Host setting in the MPPE Connect window connects to maspard. This fails with a message if no maspard is running on that machine.

Using dpumanager

The MasPar DPU job manager daemon, dpumanager, maintains the queue for the DPU, determines which programs are running on the DPU and which are swapped to disk, and ensures that the DPU state is properly initialized between jobs. The dpumanager(8) man page provides on-line details about the dpumanager.

Restarting dpumanager

dpumanager is normally started at boot time, as specified in the file /etc/rc.maspars. However, it can be restarted or changed temporarily from the command line while the system is running.

To restart the dpumanager, log in as root and follow these steps:
1. Shut down the current daemon by typing:

   # mpshutdown

2. Start up the new daemon by typing dpumanager [options]. For example, you might type:

   # dpumanager -jobs 4 -sdir /da/swap

   The above allows up to four jobs to context switch, and allows processes to be swapped out to a disk array file-system.

   When started, dpumanager puts itself in the background by forking a child process and exiting the parent.

To change the way the DPU runs on a permanent basis, edit the file /etc/rc.maspars; the parameters you place there will be used every time the system is booted. You must be logged on as root to edit this file.
Swap-to-Disk and the dpumanager

The MasPar dpu swap facility ("swap-to-disk") allows users to control how and when MasPar programs are swapped out to disk. The swapping scheme is fully automatic, yet still allows both users and system administrators to control how individual programs are treated.

You can specify parameters to the dpumanager command that specify where program swap images will be stored and determine the length of the default "slice time"—the amount of time the program has assigned to it for running and swapping. In addition, the mpctl command can be used to determine which programs are designated as "swappable," "held," or "background." mpctl is used to modify the dpumanager's default settings and to change the state of a program that is currently running.

Setting dpumanager Options

The following dpumanager options affect how the PE memory (PMem) is partitioned, the maximum time limit for all jobs, and the time allocated to a time slice. dpumanager controls context switching (DPU sharing) and is responsible for setting up where and when a job is swapped out to disk.

-jobs \( n \)  
This specifies the maximum number of programs (jobs) that can be in PE memory (PMem) and, by inference, the size of memory partitions. The default is 1 (which disables context switching), and the legal values are 1 to 16. For example, on a machine with 16 KBytes of PMem, a value of 1 creates a single 16-KByte partition, a value of 2 creates two 8-KByte partitions, and so on. For context switching, a value of 4 usually works well. Use the mpconfig command to determine how much PMem your machine has.

-maxtime \( t \)  
This specifies how long any DPU job can run (\( t \) is seconds). When a job exceeds this limit, the job manager sends the offending process a SIGXCPU signal, causing it to exit. The default is no time limit.

-nodaemon  
This prevents the job manager from putting itself in the background.

-pmem \( k \)  
This sets the default amount of PMem (in KBytes) that each job is assigned. The value you enter is rounded up to the nearest integer multiple of the partition size. The default amount is one partition of PMem. You can use the mplimit command to change the object file header to request a different amount of PMem for a specified executable.

If you enter the following command on a machine with 16 KBytes of PMem, PMem is divided into eight partitions of 2 KBytes each, but each job requests 4 KBytes (2 partitions) of PMem by default:

\[ \text{dpumanager -jobs 8} \]

\[ \text{mplimit command changes fields in the object file header of the specified executable to request a specified amount of PMem for that executable. When mplimit is not used, programs use the defaults as specified by the -jobs and -pmem options.} \]

For example, if you enter the following command on a machine with 16 KBytes of PMem, PMem is divided into eight partitions of 2 KBytes each:

\[ \text{dpumanager -jobs 0} \]

When you compile your program, a field in the object file header is set to request one partition (2 KBytes in this example:}
dpumanager -jobs 8 -pmem 4

If each job uses this default, then no more than four jobs can share the DPU at one time. If you use mplem to request a lower amount of PMem for some jobs, then as many as eight jobs may share the DPU at one time.

-sdir path

This option enables the swap-to-disk facility and specifies the directory programs will be swapped to when they are swapped out. The performance of swapping to disk is greatly affected by the device type of the swapping directory. The MasPar disk array filesystem is recommended for the best swapping performance.

-stime t

This specifies the minimum amount of time allocated to a "swap slice." (t is seconds). If the "-stime" option is not supplied, the default will be calculated based on IO operation speed for the filesystem of the swapping directory. Each program's swap to disk time period is determined by the following equation using the default time period and its PMEM partition size:

\[ \text{swap_slice} = \text{default_swap_slice} \times \text{part_size} \]

NOTE: A value of less than 3 (3 seconds) can result in thrashing.

-trace

This turns on the trace. This option is mainly for debug purposes.

-zq

This suppresses the MasPar copyright notice.

The dpumanager is responsible for starting dpui/ml and dpuswitchd. The dpui/ml daemon manages instruction memory loading. The dpuswitchd daemon provides a context for the kernel to do context switch scheduling.

Modifying dpumanager with mpctl

The mpctl command is used to modify the dpumanager's default settings "on the fly" and to change the state of a job that is known to the dpumanager. If you're logged in as root, you can change any job; otherwise, mpctl affects only your own jobs (if it is enabled).

By default, all programs are "swappable," which means that they will run until their time is expired, and then they will be swapped out to disk. You use the mpctl parameters to have a program be "background" or "held." Background programs are low priority programs, and held programs are immediately eligible to be swapped out to disk, but they are held from being swapped back in (see page 5-10 for more information).

NOTE: You can't change the flags on diagnostics programs with mpctl.

The following options can be used with mpctl:

-\( h \)

Turns on the "held" flag on a program. A currently running program will be immediately stopped and swapped to disk. Programs with the "held" flag turned on are not eligible to be swapped back in from disk.

-\( nh \)

Turns off the "held" flag.

-\( b \)

Turns on the "background" flag on a program. This increases the program's priority number and therefore moves it to a lower priority. If there are any regular programs waiting to run, the "background" job may be immediately swapped to disk.
-nb

Turns off the "background" flag.

-s

Turns on the "swappable" flag on a program. This setting is the default. If a program is "swappable" and its time period is exceeded, then when another program requires the DPU memory the "swappable" program will be swapped to disk.

-ns

Turns off the "swappable" flag. You can only change this flag if you are super user.

-limit

This parameter sets the maximum time limit (in DPU seconds) on a program. Once the time limit is exceeded the program will be killed with a SIGXCPU signal. Time spent waiting in the queue or swapped out to disk don't count against this limit. If you are not logged in as super user, the number you type for -limit cannot be greater than the system maximum time limit specified for dpumanager.

-syslimit

This changes the default system time limit value. You can only change this limit if you are super user.

-sysslice

This changes the system's default time period for the "swap slice." You can only change this limit if you are super user. Changing this limit doesn't affect already running jobs.

-swapout

This option is used to manually swap a job out to disk, even if the job does not have the "swappable" flag turned on. You can only issue this command if you are super user.

-swapin

This option is used to manually swap a job back in from disk, even if the program is "background" or hasn't used up all of its swap time period. You can only issue this command if you are super user.

-prio

Manually change the priority of a job, which changes the job's position in the queue.

Understanding Swapping Conditions

A regular program will be swapped out of memory and into a disk file when the following conditions are met:

1. The program has been running for more than the specified time period, measured as seconds of PMEM resident time since the program was started or was last swapped in.

2. The program is marked as eligible for swap to disk. Users have the option of marking a program as eligible or ineligible, with the default being eligible. Note that some programs are never eligible (see the "Fine Print" section below).

3. There is a program waiting for the DPU memory held by the current program.

4. There is sufficient space for the program's context on the disk.

5. The program is not currently holding any IORAM buffer.

Once a program is swapped to disk, its priority is recalculated before placing it in the queue list. When the swapped-to-disk program reaches the head of the queue list and is ready to run, the swapping facility will then try to swap it in from disk. The following conditions must be met for swapping in from disk:
1. The program needs to use the DPU.

2. The program must stay swapped out to disk for the specified time period if there is not enough PMEM for it without swapping other programs out to disk.

A system administrator or the owner of a program can use the command `mpct1` to mark the program so as to modify the above conditions as follows:

1. A program may be marked as "held". A "held" program will be immediately swapped to disk as long as it does not hold IORAM buffers. It will stay "in limbo" until it is marked as "not held" again.

2. A program may be marked as a "background" task. A "background" task can be running only if there are no other "non-background" tasks running. It will be immediately swapped to disk if any "non-background" program wants its memory. To prevent thrashing, background tasks must be resident in the disk for a period after they are swapped out. This minimum resident time is the same as the default slice time period.

3. A program may be marked as "unswappable." An unswappable program cannot be swapped to disk until it is redesignated as "swappable." Note that diagnostic programs are always unswappable and the "swappable" flag cannot be turned on or turned off.

**NOTE:** MPPE jobs that are swapped out to disk will remain swapped out even when you are doing data inspection. The job will only be swapped in from disk if you invoke an execution control operation (Step, Skip, Continue, Go To Line, Go To Routine).

**Swap Scheduling**

Each program is assigned a priority number based on its size, the size of its assigned time period, and its assigned properties (such as "back" or "held").

Each program may have the following three states:

- queued: the program is queued for accessing the DPU machine.
- active: the program is active inside the DPU, may or may not running.
- swapped: the program is swapped into the swapping disk.

All the programs are either placed in the active list or queue list depending on whether they are in the PMEM or not. In the active list, programs with larger priority number (such as held jobs) are placed before programs with a lower priority number. In the queue list, programs are placed in the reverse way. Therefore, the `dpumanager` first swaps to disk any low priority programs, and it first swaps in from disk any high priority programs.

The `dpumanager` enters the swapping schedule routine after every ten second clock tick or each time it receives notice of any of the following events: a program starting, a program exiting, or a `mpct1` command that changes an existing program's property (such as turning "background" on). In the swapping schedule routine, the `dpumanager` updates active programs' running information including the run time. It then checks the first program in the queue list. If it is ready to run and there is enough PMEM for it, then the `dpumanager` will start it the program or swap it in from disk. But if there is no PMEM available for it, the `dpumanager` will search the
active list for programs to swap out that will release PMEM for the ready program. After the ready program is started or swapped in, it is removed from the head of the queue list and placed in the active list. If no program in the queue list is ready to run or the queue list is empty, the dpumanager sleeps until the next event.

For "diagnostic" and "held" programs, their priority is

\[
prio = base_prio
\]

For other programs, the priority is calculated as:

\[
prio = base_prio + 2 \times (part_size - 1) + 10 \times ratio + 2 \times partitions \times swapped
\]

The base prio number is 100 for regular programs, 150 for "diagnostic" programs, 1000 for "background" programs, and 10000 for "held" programs (this number is so great that "held" jobs end up having no priority).

ratio is the program’s total PMEM resident time divided by the time used so far by the program ("live time").

partitions is the total number of PMEM partitions specified by the dpumanager; the value for partitions is always larger than or equal to a program’s part_size.

swapped is 1 if the program has been swapped to disk; if the program hasn’t been swapped to disk, swapped is 0. Thus, a program that has been swapped to disk is always placed after newly-started programs in the queue list. A program’s priority is calculated and repositioned every time when it is started, when it is swapped out, and when its property is changed by mpctl.

A regular program’s priority number is decreased by 1 if another regular program is placed ahead of it. Similarly, a "background" program’s priority number will be decreased by 1 if another "background" program is placed ahead of it. Therefore, except for "held" programs and "background" programs, none of the other programs will be blocked forever. And if all the programs are "background" ones, they compete for the DPU in the same way as regular programs do.

In general, the speed of swap to disk depends on the following factors:

- Swapping time is proportional to the machine size. Swap to disk of a 4K job on a machine of 2K PEs takes about twice as much time as swap to disk of a 4K job on a machine of 1K PEs.
- Swapping time is proportional to the job size. Swap to disk on an 8K job takes about twice as much time as a 4K job on the same machine.
- Swap to disk on the MasPar Disk Array filesystem is about 10 times faster than swap to disk on the UNIX filesystem. Also, using an 8 disk array for swap to disk is approximately 2 times faster than using a 4 disk array.
- Swap to disk on the Unix file system is about 4 times faster than swap to disk on the network filesystem.
Special Notes

- As discussed above, diagnostic programs are exempt from being swapped out. These include any program started by the hdb command, as well as certain other diagnostics. These programs cannot be affected by marking them "held," "swappable," or "background" once they have started executing.

- There is a limit to the total number of active plus swapped-out programs. The current limit is 16. Once the limit is reached, only those programs that are already assigned PMEM or disk space may continue executing until one of those programs has exited. If none of these programs are runnable, the system will stop running programs.

- During a swapping operation the DPU and the dpumanager are busy performing IO operation. This may take several minutes for large programs on large DPU machine. For the duration of this period, no programs execute and no debugging of in-memory programs takes place. Any operations requiring the attention of the job manager, including running the mpctl command and certain debugger operations, will have to wait for completion of the operation. Typing ^c may have no effect. In addition, newly submitted programs will not show up in the queue when you type mpq.

- A swapped-out program still holds all the ULTRIX resources that it had while it was active, including open files, other I/O resources, and front-end swap space.

Swap File

The swap file is stored in the swapping directory. The file contains 5 sections:

1. Header identifying file (magic, pid, uid, time, machine size, job manager’s version) and offsets of the different sections (PMEM data, CMEM data, DPU user structure, ACU user structure).

2. The user program’s PMEM data.

3. The DPU user structure including router, PREG, etc.

4. The ACU user structure including CREG, CSREG, etc.

5. The user program’s CMEM data.

Other MasPar Utilities

This section describes briefly MasPar commands that are relevant to system administration. The man pages for each of these commands contains more detailed information. To display a man page, type man command_name at the system prompt.

mpconfig

This command displays configuration information about the DPU and the front end. It must be run from a machine attached to a MasPar DPU. It generates a display similar to the following:

MasPar DPU Model MP-110Z (32 rows, 64 columns)
ACU IMEM size: 1 MByte
ACU CMEM size: 128 KBytes
PE memory size: 64 KBytes
PVME in I/O slot #2 (32 MBytes)
MPDA unit #0: 8 data drives of type DK516-15
mpfile
This command displays the type of a given file. It is especially useful in distinguishing MasPar Object File Format (MOFF) files from other executable files. It also performs the functions of the file command.

mpi[hostname]
This command prints out information about all DPU's on any attached local area network supporting SO_BROADCAST sockets and the broadcast address INADDR_BROADCAST.

mplimit
This command prints and sets the DPU usage limits stored in a MOFF executable file. These limits tell dpumanager what the expected resource requirements are for the program.

mpq[hostname]
This command examines the DPU job queue and reports the current rank in the queue for each job. Job status definitions are as follows:

- Active: Currently holds the DPU.
- Waiting: Waiting for access to the DPU.
- Inactive: The DPU device is open, but access has not yet been requested, or the device has been released.

mpshutdown
This command sends a termination signal to the DPU job manager daemon, dpumanager. When the job manager receives this signal, it sends a hang-up signal to all pending DPU jobs and then exits. You must have root privileges to use this command.

mpsizer
This command prints the decimal number of bytes required by the MasPar program, including front end text, FE data, FE bss, ACU text, ACU data, ACU bss, PE data, and PE bss portions. If no output file is specified, a.out is used.

mpstat
This command prints the MasPar job accounting statistics accumulated by the DPU job manager. Following are the options:

- `-a` print chronological list of all relevant jobs
- `-c` clear the accounting file (must have write permission)
- `-f filename` use specified file instead of /usr/adm/dpuacct
- `-s` print summary (default if -a or -c is not specified)
- `-t n` display n most recent events (tail option)
- `-u username` limit display to entries for specified user
- `-q` suppress printing of the MasPar copyright notice

mpswopt
This command prints or sets the PE-array size requirements stored in a MOFF executable file.

mptimelimit
This command sets the maximum time limit for all MasPar DPU jobs. This limit remains in effect until the next mptimelimit command or until the job manager is restarted. The command is equivalent to the -maxtime argument on the dpumanager command.

mpzap
This command enables you to change certain defaults that might not be appropriate to the physical environment where software will be installed. These defaults can pertain to MP-1 hardware, or to such things as yellow pages service numbers in a complex network.
mp2ap is especially useful in changing the port number used by maspard and mplicensed, and in tailoring the MasPar Fortran compiler to run on systems with more than 1,024 PEs.
Chapter 6

MPDA System Administration

This chapter provides procedures for bringing up the MPDA, initializing the DA controller and the disk banks, and other administrative procedures. Further detail is available in the chapter "Guide to Disk Maintenance," in the DEC manual, System and Network Management, Volume 2. For standard MP-1 system administration procedures, refer to the MasPar MP-1 Installation Manual.
Operating System Configuration

Before you can use the MPDA, it is necessary for the disk array to be configured properly in the operating system kernel. Use the following procedure to assure proper operating system configuration. This procedure presumes that the MasPar system software has already been installed according to the instructions in the current MP-1 Release Notes.

It may be necessary to rebuild the operating system twice as part of this procedure. However, if you find yourself repeating steps without making any progress, it is time to call MasPar Customer Support.

1. Make sure the disk array is fully powered on and connected. If you change anything, reboot the workstation.

2. Check the operating system configuration with the following command:

   ```shell
   # /etc/uerf -R -r 300 | more
   ```

   This causes the latest system startup log message to be displayed. Look for the following line:

   ```text
   vba0 at slot 2 (3VIA/MVIB)
   ```

   This is for the VMEbus Adapter. The slot number is not significant. If you don’t see this line, something is seriously wrong. Check your bearings. Call MasPar Customer Support.

   Shortly after the "vba0" line you should see something like:

   ```text
   ms0 at vba0 csr ...
   ```

   This is for the MPDA controller. If you don’t see this line, either the operating system is not configured for an MPDA at all, the hardware has not been installed correctly, or something is broken. Make sure everything is plugged in on the VMEbus and the cables are properly attached to the workstation; if you find something wrong, return to step 1. Otherwise, continue on with this step.

   Immediately after the "ms0" line you should see a line like the following:

   ```text
   da0 at ms0 slave 0
   ```

   If you have more than one MPDA bank, there should be one line like this (with different da and slave numbers) for each bank. If these lines are missing, most likely the disk array was not configured correctly before the current operating system was built, but there may also be hardware problems. Check the text under "Other Errors" later in this section. If there are hardware problems, correct them and go back to step 1. If the da lines are missing, go to step 2.

   If you got this far, the operating system itself is properly configured and you can skip to step 5.

3. Make sure the MPDA controller is properly configured. Use the procedures described under "Verify the Hot Standby Status" and "Verify Bank Status," below. If you change anything, go back to step 1. Otherwise, continue with step 4.

4. Rebuild the operating system. A procedure for rebuilding the operating system, using "/genvmunix" as a base, are in the MasPar Installation Guide. After you have done that, go back to step 1.

5. Make sure the proper device files are present, as described under "No Device Entry," below.
The system software should now be configured correctly.

**Establishing MPDA Communications**

**NOTE:** The following examples show a DA3004 without the hot standby option configuration.

Begin by using the `dastat` command to make sure the MPDA has been installed correctly, and verify that the kernel can communicate with the MPDA. Type:

```
# dastat
```

You see the following information on screen:

- Physical addressing
- Disk logging is enabled
- OS logging is enabled
- Bank 1 standby serves all banks
- Format interleave is by track
- Reconstruct interleave is by track

- Retries = 16
- Data Chan TMO = 5
- Ready TMO = 255
- Spin up delay = 2
- Status filter = 0000
- Monitor = 0000
- Buffer presence = -0-0-0-00-

If the `dastat` command fails, check the following areas:

**No Device Entry**

Verify the device files are present:

```
# ls -l /dev/rda*
```

If not present, "make" the device:

```
# cd /dev
# ./MAKEDEV da0
```

Then repeat the device verification:

```
# ls -l /dev/rda*
```

**Other Errors**

- If the message does not contain a reference to `ms0` after the line
  `vba0 at slot 2 (3VIA/MVIB)`
  the SVMII board is not being properly accessed. The problem could be an improperly seated 6U adapter or SVMII PCB or something similar.

- The message
  `da0: da_dmaisr has unexpected value = 0xff`
  indicates that the SVMII interrupt status register is not coming up properly after a reset.
• The message
dawaitcmd: CMDBUSY in PSR won’t clear
usually indicates a cabling problem between the DPU and the MPDA. It could also
indicate that the MPDA is powered down.

• The message
dslave: CONTROLLER WON’T SYNC
indicates some type of initialization problem with the DA controller. Try RESEtIing
the DA controller and rebooting the system.

Verify the Hot Standby Status

To make sure the controller is configured correctly, type:
dastat

Verify that the standby disk mode is correct. If the line:
Each bank has standby
appears instead of:
Bank 1 standby serves all banks
at the prompt, enter:
# dainit -ml /dev/da0

to set the configuration so that one standby serves all banks.

NOTE: Whenever you change the DA controller or bank configuration, you must reboot the
front-end workstation.

Verify Bank Status

Use dabstat to check the bank configurations. This example checks Bank 1:
# dabstat -bl

<table>
<thead>
<tr>
<th>Drive ID</th>
<th>Sub ID</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>0</td>
<td>4 data drives</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use primary system area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check drive array addresses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log soft and hard errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recover policy is R/E/P</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBS/DSBL</th>
<th>Bank Status</th>
<th>Total blocks</th>
<th>Reconstruct Address</th>
<th>Drive status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>On line, Mounted -0-0-0-0-0-0-</td>
<td>682920</td>
<td>-1</td>
<td>00 30 00 30 00 30 00 30 30 00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive controller not present or buffer fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive online/mounted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive controller not present or buffer fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive online/mounted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive controller not present or buffer fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive online/mounted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive controller not present or buffer fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive online/mounted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drive online/mounted</td>
</tr>
</tbody>
</table>
Drive 10  Drive controller not present or buffer fault

("drive 10" is absent unless it is the standby disk.)

See Table 6-1 for the drive status codes.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>DRIVE STATUS CODE</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0</td>
<td>Buffer fault or not present</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>Drive not assigned *</td>
</tr>
<tr>
<td>Initialization Error</td>
<td>1 1</td>
<td>Drive not selected (drive not present or unit select switch wrong)</td>
</tr>
<tr>
<td></td>
<td>1 2</td>
<td>Drive is busy (reserved by other channel)</td>
</tr>
<tr>
<td></td>
<td>1 3</td>
<td>Drive does not become Ready (check drive)</td>
</tr>
<tr>
<td></td>
<td>1 4</td>
<td>Drive fault does not clear (check drive)</td>
</tr>
<tr>
<td></td>
<td>1 5</td>
<td>Drive seek error does not clear or on-cylinder not set (check drive)</td>
</tr>
<tr>
<td></td>
<td>1 6</td>
<td>Drive Read_Clock not valid (check drive)</td>
</tr>
<tr>
<td></td>
<td>2 7</td>
<td>Error reading System Area (drive not formatted)</td>
</tr>
<tr>
<td></td>
<td>2 8</td>
<td>System Area not valid (drive must be reformatted)</td>
</tr>
<tr>
<td>Warning (OnLine)</td>
<td>2 9</td>
<td>System Area bank/drive address position doesn't match (drive cables wrong)</td>
</tr>
<tr>
<td>Mount Success</td>
<td>3 0</td>
<td>Drive Online/Mounted</td>
</tr>
<tr>
<td>Mount Success</td>
<td>3 1</td>
<td>Drive Online/Mounted (write-protected)</td>
</tr>
</tbody>
</table>

Table 6-1  Drive Status Codes

NOTE: "Drive not assigned" only occurs for the standby drive when it is configured in Standby Mode 2 (see control byte of controller configuration) and it is not assigned to this bank.
Setting Up File Systems

The procedures below describe how to set up your file systems on the disk array. Refer also to "Guide to Disk Maintenance," in the DEC manual System and Network Management Volume 2.

**NOTE:** The h partition is reserved for diagnostic use. The MPDA diagnostics use this partition as a scratch area. The program chpt(8) will not modify partition h or modify any other partition so that it overlaps partition h.

1. Check the default file partitions, as in the following example:

2. **Example:** chpt -q /dev/rda0a

3. Run /etc/newfs

Syntax for newfs(8) command:

/etc/newfs [-v] [-n] special-device device type

**Example:** newfs -v -n /dev/rda0a mpda-515-4

For the MasPar Disk Array, device type reflects the number of data disks:

mpda-515-4

mpda-515-8

Default Partitions

This section shows examples of the default partitions for four- and eight-disk 516s. All the following examples are from /etc/chpt -q /dev/rda0a.

**MPDA-516-4**

<table>
<thead>
<tr>
<th>partition</th>
<th>bottom</th>
<th>top</th>
<th>size</th>
<th>overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>2668799</td>
<td>2668800</td>
<td>b,c,f</td>
</tr>
<tr>
<td>b</td>
<td>2668800</td>
<td>5337599</td>
<td>2668800</td>
<td>a,c,f</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>10675199</td>
<td>10675200</td>
<td>a,b,d,e,f,g</td>
</tr>
<tr>
<td>d</td>
<td>5337600</td>
<td>8006399</td>
<td>2668800</td>
<td>c,g</td>
</tr>
<tr>
<td>e</td>
<td>8006400</td>
<td>10675199</td>
<td>2668800</td>
<td>c,g</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>5337599</td>
<td>5337600</td>
<td>a,b,c</td>
</tr>
<tr>
<td>g</td>
<td>5337600</td>
<td>10675199</td>
<td>5337600</td>
<td>c,d,e</td>
</tr>
<tr>
<td>h</td>
<td>10675200</td>
<td>10684800</td>
<td>9600</td>
<td></td>
</tr>
</tbody>
</table>

**MPDA-516-8**

<table>
<thead>
<tr>
<th>partition</th>
<th>bottom</th>
<th>top</th>
<th>size</th>
<th>overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>5337599</td>
<td>5337600</td>
<td>b,c,f</td>
</tr>
<tr>
<td>b</td>
<td>5328000</td>
<td>10665599</td>
<td>5337600</td>
<td>a,c,f</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>21350399</td>
<td>21350400</td>
<td>a,b,d,e,f,g</td>
</tr>
<tr>
<td>d</td>
<td>10675200</td>
<td>16012799</td>
<td>5337600</td>
<td>c,g</td>
</tr>
<tr>
<td>e</td>
<td>16012800</td>
<td>21350399</td>
<td>5337600</td>
<td>c,g</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>10675199</td>
<td>10675200</td>
<td>a,b,c</td>
</tr>
<tr>
<td>g</td>
<td>10675200</td>
<td>21350399</td>
<td>10675200</td>
<td>c,d,e</td>
</tr>
<tr>
<td>h</td>
<td>21350400</td>
<td>21369599</td>
<td>19200</td>
<td></td>
</tr>
</tbody>
</table>
Mounting Partitions as MPFS

MPDA partitions should be mounted as file system type "mpfs" instead of the normal "ufs" used for regular disks. The MPFS mount command takes the same options as the UFS mount command, as documented in the mount(8ufs) man page, except that `-t mpfs` must be added to the command. For example,

```
mount -t mpfs /dev/da0a /da
```

would mount the a partition of the disk array as type mpfs on the directory /da.

MPFS file systems may be configured in the /etc/fstab file. The entry is identical to the equivalent UFS entry, except you must substitute mpfs for ufs. For example:

```
/dev/da0a:/da:rw:1:4:mpfs::
```

With this entry, either of the following commands will mount /da:

```
mount -a -t mpfs
mount /da
```

The `fsck(8)` command treats MPFS file systems the same as UFS file systems. The single command `fsck -p` will check all UFS and MPFS file systems listed in /etc/fstab.

The MPFS file system, which uses 256Kbyte blocks, behaves similarly to the UFS file system for most purposes. Refer to the `mpfs(5)` man page for more information.

NOTE: MPDA partitions may be mounted as file system type UFS provided that the partition size is less than 2 giga-bytes (4,194,303 sectors maximum) and the file system block size is 8192 or 4096 bytes. However, UFS will not provide as high performance as MPFS when using parallel I/O functions to access MPDA files.

Disabling a Drive

Normally, disabling occurs automatically. To manually disable a drive, use `dadsbl(8)` to disable a specified drive in a specified bank:

```
dadsbl -b1 -d4 /dev/da0
```

This example disables drive #4 in bank 1:

```
CAUTION:
If there is already a drive assigned to the disable field, do not disable another drive on the same bank.
```
Assigning a Standby Drive

In the case of an error, the software will automatically substitute for a drive if there is a hot-standby drive present. If you would like to manually substitute the stand-by for a drive, use the dasubs(8) command.

In this example, the standby drive is being substituted for drive 6.

dasubs -b1 -d 7 /dev/da0

```
# dabstat -bl

Drive ID    34   DK515-78
Sub ID      0
Mode
  Use primary system area
  Check drive array addresses
  Log soft and hard errors
  Recover policy is R/E/P
SUBS/DSBL   6 / 6
Bank Status On line, Mounted -0-0-0-000
Total blocks 682920
Reconstruct Address -1
Drive status 00 30 00 30 00 30 00 30 00 30 30 30
  Drive 1: Drive controller not present or buffer fault
  Drive 2: Drive online/mounted
          
Drive 10: Drive online/mounted
```

Notice that both the SUBS and DSBL fields contain drive 6. At this point, the standby drive IS NOT being used. To finish the substitution process, a darecon must be run to reconstruct the data from the disabled drive onto the standby drive.

darecon -b1 /dev/da0

After the reconstruct has completed, run dabstat to verify that the DSBL field has cleared.

```
# dabstat -bl

Drive ID    34   DK515-78
Sub ID      0
Mode
  Use primary system area
  Check drive array addresses
  Log soft and hard errors
  Recover policy is R/E/P
SUBS/DSBL   6 / 0
Bank Status On line, Mounted -0-0-0-000
Total blocks 682920
Reconstruct Address -1
Drive status 00 30 00 30 00 30 00 30 30 30
  Drive 1: Drive controller not present or buffer fault
  Drive 2: Drive online/mounted
          
Drive 10: Drive online/mounted
```
Reconstructing Data

Reconstruction is a process that rebuilds the data on a drive using the data from other drives. The sole purpose of the parity drive is to support the reconstruction process.

Reconstruction can happen automatically, or it can be initiated manually when a failed data/parity drive is replaced by using the `darecon(8)` command. The following is a summary of the usage for `darecon`:

```
darecon -b bank [-t target] device
  -b     bank—Legal values are 1-4
  -t     target—Legal values are 0-1.
```

Use the target field only when DSBL not equal zero, SUBS not equal zero, and DSBL not equal SUBS. If this is the case,

```
-t0    DSBL is the reconstruct target
-t1    SUBS is the reconstruct target
```

Reconstruction is not possible unless a drive has been assigned either as a substitute drive (SUBS) or is disabled (DSBL).

The following table shows the action taken by `darecon` based on the value of the DSBL and SUBS fields, X = 1-10, Y = 1-9. (The DSBL and SUBS fields show the disk status before `darecon` has been run)
<table>
<thead>
<tr>
<th>DSBL</th>
<th>SUBS</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>Reconstruct on drive X. After reconstruction is done, DSBL = 0.</td>
</tr>
<tr>
<td>0</td>
<td>Y</td>
<td>Reconstruct on drive Y. After reconstruction is done, SUBS = 0.</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Reconstruct on drive selected in the target field. After reconstruction is done, the field selected (DSBL or SUBS) = 0.</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Reconstruct drive Y onto the standby drive. After reconstruction is done, DSBL = 0 and SUBS = Y.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Normal System. Reconstruction not possible.</td>
</tr>
</tbody>
</table>

Table 6-2  Actions Taken by Reconstruction

Installing a Replacement Disk Cannister

1. Verify that there is a drive already disabled or substituted for:
   
   ```bash
dabstat -bn
   ```

2. Replace the disk canister.

3. Spin-up the drive by typing:
   
   ```bash
damount -bn -d0
   ```

   **NOTE:** It takes 45 to 60 seconds for the drive to spin-up.

4. Verify that the drive has spun-up by typing:
   
   ```bash
dabstat -bn
   ```

5. If the drive is not a status 29 or 30 then type:
   
   ```bash
damount -bn
   ```

6. Verify that the drive has mounted:
   
   ```bash
dabstat -bn
   ```

7. To normalize system area, type:
   
   ```bash
daformat -bn -f0
   ```

8. Verify that the drive is on line and mounted by typing:
   
   ```bash
dabstat -bn
   ```

9. Reconstruct the data:
   
   ```bash
darecon -bn
   ```
This chapter identifies some common problems that you or other users might encounter when using the MasPar® MP-1. The chapter presents tips for preventing these problems, and suggestions for solving them.
Problems with DPU Access

Users might report that their stations, terminals, or a window are hung, with the job manager (dpumanager) appearing to be in an infinite loop.

Accessing the DPU

ULTRIX regards the DPU as a hardware device with a device driver and entry points in /dev/dpu. The job manager program, /usr/maspar/etc/dpumanager, is started at boot time. It requires root privileges to start.

When the job manager starts, it loads two files into the DPU: the microcode and the Array Control Unit (ACU) kernel. The ACU kernel provides protection for the multiprogramming environment (context switching) and I/O operations in the DPU. All DPU programs require the ACU kernel to run. Currently, there are no interfaces to this kernel that are accessible to customers.

If you boot the DECstation with the DPU disconnected or powered down, ULTRIX comes up, but the DPU is ignored, as if it were not there. Therefore, the DECstation can run while the DPU is completely nonfunctional.

The DPU job manager continually runs diagnostics and post-mortem checks, and writes results to this log file: /usr/adm/dpujobmgr.log. Examining this file can provide clues to problems you might experience with the job manager.

Determining if the DPU Is Running

There are several ways to determine if the DPU is running:

- If the DPU job manager has been halted with a kill command, mpq and mpconfig might erroneously report that the job manager is running. To definitively find out if the job manager is running, move to a shell window and type:

  `ps -ax | grep dpu`

  The output should include three daemon processes: dpumanager, dpuswitchd, and dpuml.d.

- The DPU displays messages on the screen when the DECstation is being booted, just like any other device. These include:

  `vba0 at slot 2`
  `acu0 at vba0 csr 0xfc000000 vmea32d32 vec 0xd0 priority 1`

  A copy of these messages is written to the ULTRIX system log file, which can be read by typing:

  `/etc/verf -R -r 300 | more`

  Look at that file to see if these messages from the DPU are there. If not, your DPU is not connected properly.

- After the system has booted, you can use the commands mpq and mpconfig to see if the software and hardware (respectively) are running, as follows:
To see a list of the DPU jobs running, enter:

    mpq

or

    mpq hostname

To get a report on the DPU hardware, enter:

    mpconfig

If the DPU is not online when you try to start up a DPU user process (an MPL program, for example), you will see the message:

Unable to open the DFU device files

Starting the Job Manager

If the DPU job manager is not running, login as root, then enter:

    dpumanager -jobs n [other_options]

If the DPU job manager is still not running, make sure the DPU is powered up, then reset the DECstation.

Refer to Chapter 5 for details on the job manager.

Checking the DPU Connections

If the DPU is powered up but still does not respond to the DECstation, make sure the DPU is properly connected to the DECstation by following these steps:

1. Turn off the DPU.
2. Consult the MasPar MP-1 Installation Manual and make sure that all of the connections between the DPU and the DECstation are fastened securely.
3. Power up the DPU.
4. Reset your DECstation.

Reporting Problems

If you continue to have problems, contact MasPar Customer Support at the numbers and addresses shown in the preface. Provide the following information with your defect report, as applicable:

- The output from mpconfig.
- The system error log. Type `/etc/uerf -R` to get a reverse listing of recent events. Type `/etc/uerf -R -o full` to extract the log of events around a crash.
- Relevant portions of the DPU job manager log, `/usr/adm/dpujobmgr.log`. You may need to examine the entire file or just the tail, depending on the size of the file.
- The DPU accounting file, `/usr/adm/dpuacct.log`. See the `mpstat` man page for descriptions of the options you can use to get different kinds of information in this file. For example, `mpstat -s` gives a summary of all jobs run since the DPU
job manager was restarted, `mpstat -a -t num` gives a chronological accounting of the last `num` jobs run, and `mpstat -c` clears the accounting file.

License Manager Problems

Access to many MasPar products is controlled by a license manager. If you try to use a product for which you do not have a license, you will get an error message. Contact your MasPar Sales Representative to purchase additional licenses.

License Limits

When any licensed product (such as MPL, MasPar Fortran, or MPPE) executes, it checks with the license manager software (`lmgrd` and `mplicensed`) for the number of users currently using the program. If the licensing limits are already met, the execution attempt fails, and the license manager tells the user that the system is not licensed for that many users. The language compilers and MPPE put you in a queue while waiting for a license to become free. To get out of this blocked state, enter `<Ctrl>-C` at the prompt or in the MPPE Stdin/Stdout window.

License Failures

If you get any messages from the license manager, such as `license failure = -5` after executing a license-protected program, check the log file, `/usr/maspar/etc/lmgrd.log`.

Conflicting Licenses

The MasPar software is controlled by Version 1.5 of the Highland license manager. If you have other software products that also use the Highland license manager, it might cause problems in recognizing your MasPar product licenses.

You can merge your license files provided the other licensed products also use Version 1.5 of the Highland license manager. In the merged `license.dat` file you should keep one of the `SERVER` lines, all of the `DAEMON` lines, and all of the `FEATURE` lines. Replace all your old `license.dat` files with this merged one. Then restart the license manager process using `lmstart`.

Reporting Problems

If you continue to have problems, contact MasPar Customer Support at the numbers and addresses shown in the preface. Provide the following information:

- The version of the license manager you are running with your other products, if applicable.
- The error messages you receive.
Compiler Problems

The MasPar compilers return error messages that identify programming errors and usage errors. Users might report some compiler problems that are not attributable to simple programming errors. By turning on "compiler logging," you can gather detailed information about the compile that might aid in determining the problems.

Logging Compiler Activity

When enabled, the compiler logging feature creates a record every time someone uses a MasPar compiler. Its default status is "disabled". It puts the records in the file $MP_PATH/log/mpcompile.log. To enable compiler logging:

1. Create or edit the file $MP_PATH/log/.mplogcontrol.
   • If the file exists, change the string off to on. Make sure this file contains only the string on.
   • If no file exists, create one, and insert the string on.

2. Make sure the file $MP_PATH/log/mpcompile.log exists and that the permission bits for it and the directory $MP_PATH/log give everyone write permission.

3. Read the output in the file $MP_PATH/log/mpcompile.log. When a compile is run, this file creates a record similar to the following:

    ===== start of record
    time (GMT): Fri Apr 20 00:47:16 1990
    User ID: 296
    system/serial #: *unknown*
    hostname: songbird
    software revision: mpl.00.00
    MP_PATH: /u2/maspar/build
    command line: mpl -o bmaps.m
    error summary: exit code= 0
    time (GMT): Fri Apr 20 00:47:25 1990
    ===== end of record

To disable the compiler logging feature, edit .mplogcontrol, changing the string on to off.

For each compile record, the beginning and ending time stamps provide a record of how long the compile took. The software revision is extracted from the file $MP_PATH/.id.

NOTES: If you have turned on "compiler logging," the permissions for the file $MP_PATH/log/mpcompile.log must be set so that the owner, group, and world can read and write the file (chmod filename 666). If the permissions do not allow read-write access to all, a user's compile will fail with the message "permission denied."

If you use compiler logging, be aware that the mpcompile.log file grows very fast. Turn logging off as soon as possible, or delete and recreate the log file periodically to avoid running out of disk space.

For more information on troubleshooting compiler problems, refer to the documentation for the appropriate compiler.
Reporting Compilation and Execution Problems

If you continue to have problems, contact MasPar Customer Support at one of the numbers or addresses listed in the preface. When reporting a defect against a compiler driver, include as much information as possible about the problem, such as:

- The output from mpconfig. If you email the information from mpconfig to Customer Support, then they'll know exactly what type of system you're running on.
- The version number of the compiler driver. This is displayed when you invoke the compiler (without using the -vq option).
- A copy of the makefile or other script that runs the program.
- A record of any error messages from the compile.
- A copy of any applicable input and output.
- A copy of anything else the program uses that was not provided by MasPar (such as internally developed libraries).
- The smallest portion of your program that still exhibits the defect.

MPPE Problems

Problems when using MPPE could be related to the program being debugged, your environment, the maspard daemon, or MPPE itself.

Incorrect DISPLAY Variable

If a user tries to invoke MPPE on a workstation not normally used by that user, MPPE might appear to not respond. This could be due to the user's DISPLAY environment variable being incorrect. Check that the user's DISPLAY environment variable points to the correct workstation. For more information on setting the DISPLAY variable and on using MPPE, refer to the MPPE User Guide.

Incorrect Host Name

MPPE will fail if the system host name does not appear in the /etc/hosts file, because the maspard daemon will not run.

Incorrect User ID

If you receive the message no /dev/tty for interactive input in the Stdin/Stdout window, your user IDs on the two systems are probably different.
Incorrect System Date

If MPPE will not start up, check to make sure the system date is correct. If it isn’t, run `date (1)` as root.

Reporting MPPE Problems

If you continue to have problems, contact MasPar Customer Support one of the numbers or addresses listed in the preface. When reporting a defect against MPPE, include as much information about the problem as possible, such as:

- The output from `mpconfig`.
- The version number of the MPPE. This is displayed when you invoke the MPPE.
- The contents of `/usr/adm/maspard.log`.

If you are reporting a problem with debugging a program (as opposed to a problem with some other part of the MPPE, such as moving the MPPE windows to the front with the `Front` button), also include information about the program you are debugging.

Troubleshooting the Swap To Disk Facility

The followings are the answers to the most common questions about the swap facility.

Q: How do I know the dpu swap facility is enabled on the MasPar system?

A: Type `mpctl`. If the dpu swap facility is enabled, you should see the message `Swap facility: enabled. (The mpctl command also provides other information, such as the machine size, swapping directory, swap slice time, and so on.)`

Q: When will my job be swapped out to disk?

A: A job will be swapped to disk when all of the following conditions satisfied:

- There are jobs (either newly started or swapped-out jobs) that want the PMEM your job is using.
- Your job is marked as swappable.
- Your job has used all of its swap slice time.

Q: Is it possible to prevent my critical job (for example, a benchmarking job) from being swapped out to disk?

A: Yes, if you are logged on as super user.

- You can use the command `mpctl -ns <pid>`, where `<pid>` is the process ID of your job, to mark your job as not swappable.
- You can type `mpctl -swapoff` to totally disable the swap facility.
Q: Why are some jobs not swapped out?
A: Any of the following reasons may cause the jobs not swap.
   • The swap facility is not enabled.
   • The job is diagnostic job. Diagnostic jobs are not swappable and will run until completed.
   • There is not enough disk space.
   • The job is currently holding IORAM. Any job holding IORAM buffers cannot be swapped out until it releases all the IORAM buffers.
   • The job has been marked as not swappable.

Q: Why hasn’t my swapped-out job been swapped back in?
A: there are several reasons your job may not be swapped in:
   • Some currently running jobs may be marked as not swappable; if so, your job won’t be swapped in until those jobs are either completed or are marked as swappable. (Remember, diagnostic jobs are always marked as not swappable.)
   • There are jobs currently running and your job’s resident time on the disk has not reached its total swap slice time yet.
   • There is no disk space for currently running jobs to be swapped out.
   • Someone logged in as super user may have used mpctl -h pid to have your job held in the disk permanently. Your job won’t be swapped back in until your job is marked as a regular job again.
   • Someone logged in as super user may have used mpctl -swapoff to disable the swap facility. Your job won’t be swapped in until the swap facility is enabled again.

Q: If there are a lot of jobs in the queue, can I change my job’s position in the list so that my job can be started quickly?
A: Yes. If you log in as super user, you can use the command mpctl -prio <pri> pid to change your job’s priority, and move your job to top of the queue list.

Q: What is my job’s swap slice time?
A: Your job’s swap slice time is proportional to the size of your job. The formula for calculating the swap slice time for your job is to multiply the default swap slice time by your job size and divide that by the smallest job size. For example, say that the default swap slice time is 92 seconds, your job size is 64K, and the number of PMEM partitions is 8; your job’s slice time is then 92 * 64K / 8K, which is about 13 minutes. Type mpctl to see the default swap slice time, as well as number of PMEM partitions. The default swap slice time is calculated based on machine size and whether the swap directory is a disk array file system.

Q: Why does mpctl seem to hang sometimes?
A: mpctl does not really hang. If the dpumanager is swapping a job out to disk, this may take up to a couple of minutes for a large job. During that time, all the commands (including mpctl) that need the dpumanager have to wait for the dpumanager to finish the swap.
Q: What are the advantages of having swap to disk enabled?

A: Some of the main advantages are:

- swap to disk allows more jobs to share the system resource. Right now, up to 16 users can share the DPU machine.

- you can avoid having a single large job hog the whole system. In previous software releases, even if the job wasn’t doing anything (for example, if it was waiting for some terminal input) it would still take all of the DPU resources, and all other jobs would have to wait for machine access.

- all jobs’ queue times are shortened. In previous releases, if there were jobs hogging the DPU, all other jobs would have to wait until those jobs exited the DPU. So if somebody has a long-running large job, no other jobs can be started. This was especially a problem for some interactive small jobs. When the swap facility is enabled, these small interactive jobs are able to run without waiting for large long-running jobs to exit.

- since jobs are not swapped in if they do not need DPU, the system utilization is improved.

Q: What are the disadvantages of having swap to disk enabled?

A: Some of the main disadvantages are:

- There is overhead in the swapping operation. This overhead can be serious if there is no disk array to swap to. Even with an MPDA, the largest job (64K jobs on 16K PE machine) will take between one and two minutes.

- Since some jobs are no longer in the DPU the whole time they are running, users may find that those jobs take more time to complete once they are started. (However, it may take much less time to reach completion if you calculate from the time the job was entered into the queue; since jobs no longer wait in the queue for all previous jobs to finish, the overall time from submission is much less.)
Index

ACU kernel 1-1, 7-2
Administration
  See also System administration
ADVANCED Installation menu
    option 2-3
apropos ii
apropos 2-8, 2-11
attached monitor 2-3

big_endian 4-3
black and white monitor 2-13
Bootstrap messages 7-2

catman 2-8, 2-11
CDROM
  booting up from 2-3
CDROM problems 2-5
CDROM reader
    confirming SCSI device number 2-2
    inserting CDROM into housing 2-3
Checklist 3-6
color monitor 2-13
Communications, MPDA 6-3
Compiler drivers 3-7
Compiler logging 7-5
Configuring modem software 3-4
Configuring your system 2-2
console type 2-13
Daemons 5-1

DEC documentation v
DEC software upgrade 2-5
DECstation
documentation iv
DECstation 5000 1-1
default file system layout 2-3
Defaults, DPU manager 5-7
Directory trees 1-2
  multiple 1-3
DISPLAY variable 3-6, 7-6
  setting 3-6
doconfig command 2-6
doconfig utility 2-8, 2-12
Documentation, DECstation iv
DPU 1-1
  job duration 5-7
  job manager 7-2, 7-3
  log file 7-2
  operating system 1-1
  problems 7-2
  set up 2-8, 2-12
DPU accounting file 7-3
DPU manager 7-3
DPU manager log file 7-2, 7-3
dpuacct 7-3
dpuinit 3-9
dpujobmgr.log 7-2, 7-3
dpumanager 5-6, 5-7, 7-2

Environment variables
  DISPLAY 3-6, 7-6
  MP_PATH 3-5, 3-6

Index  1
Error messages 7-5
/etc/fstab 3-6
/etc/getty 3-3
/etc/hosts 3-2, 3-9
/etc/rc.local 3-2
/etc/rc.maspars 5-6
/etc/services 3-7
/etc/ttyS 3-3, 3-4
/etc/uerf 7-2, 7-3
/etc/services 3-9
Examples ii

File .mplogcontrol 7-5
File mpcompile.log 7-5
file systems 4-1
Font conventions iii
front-end memory mapping 4-3
genvmunix 2-8, 2-12
getty 3-3
gfi 4-4
greyscale monitor 2-13
Group maspar 1-3

Help ii
Highland license manager 7-4
Host name 3-7, 3-8, 7-6
network environment 3-2
stand-alone environment 3-2
updating 3-2
hosts 3-2, 3-9
Hot standby
See also Standby disk

Installation Guide 2-1
installation menu 2-3
Installing
before you begin 2-2
MasPar software upgrade 2-9
Installing optional products 2-1
IORAM file system 4-4
See IORAMFS
IORAMFS
exporting to NFS 4-2
mounting 4-4

Job manager 7-3
problems 7-2
Job manager, DPU 5-6, 7-2
Job swapping 7-2

Kernet 1-1
Kernel configuration file 2-4

License daemon 5-2
license.dat file 2-13
License limits 5-5, 7-4
License manager 2-2, 5-2
conflicts 7-4
failures 7-4
Highland 7-4
limits 7-4
problems 7-4
utilities 5-5
License manager daemon 5-2
license.dat 3-9, 5-2, 5-4, 3-9
Licensing data file 5-2
Link 3-7, 3-8, 3-9
little_endian 4-3
lmconv 5-5
lmgrd 5-2, 7-4
lmgrd.log 5-2
Imhostid 5-5
lmist 5-5
Imremove 5-5
lmstart 5-5
Log file, DPU 7-2
Log files 7-2
dpujobmgr.log 7-2, 7-3
lmgrd.log 5-2
mpcompile.log 7-5
system error log 7-3
Logging compiler use 7-5

man ii
Mandatory upgrade subsets 2-4
maspar
  group id 1-3
  user id 1-3
MasPar Architecture Specification iii
MasPar Commands manual iii
MasPar file system 4-2
See MPFS
MasPar Fortran 5-14
MasPar Fortran User Guide iv
maspar login account 1-3
MasPar software
  installing upgrades 2-9
  loading from scratch 2-6, 2-10
MasPar System Overview manual iii
masp ard 3-7, 3-8, 3-9, 3-10, 5-6
mdinfo 3-7, 3-8
Modem
  configuration 3-4
  configuring software for 3-3
  software configuration 3-4
  using 3-4
Modem switch
  set to CONNECT  3-3, 3-4
monochrome monitor  2-13
Motif  1-2, 2-7, 2-11
Mount  3-7, 3-8, 3-9
Mount points  3-8, 3-9
Mounting /usr/kits disk partition 2-5
Mounting file systems  4-2
MP file system  4-2
  See MPFS
mpcompile.log  7-5
mpconfig  5-7, 5-12
mp_create_license  5-5
MPDDL Reference manual  iv
mpfile  5-12
MPFS
  exporting to NFS  4-2
    size limitations  4-2
mpi  5-13
MPIPL Reference manual  iv
MPL Reference manual  iv
MPL User Guide  iv
mplicensed  5-2, 7-4
mplimit  5-7, 5-8, 5-13
.mplclogcontrol  7-5
MPML Reference manual  iv
MP_PATH variable  3-5
MPPE  3-7, 3-8, 7-6
  remote operation  5-6
MPPE User Guide  iv
mq  5-13
mpshutdown  5-13
mpstat  5-13
mpswopt  5-13
mptimelimit  5-13
mpzap  5-13
Multiple directory trees  1-3
Multiple versions of software  1-3
netsetup  3-2
Network environment  3-2
New products  2-2
New user  3-6
NFS  3-7, 3-8, 3-9
  exporting ioramfs  4-2
  exporting mpfs  4-2
nfssetup  3-7, 3-8
nodes  4-3
Notation conventions  iii

Operating system, DPU  1-1
OSF Motif  1-2

O_BIGENDIAN  4-3

Partition memory, PE  5-7
Partition sizes  5-7
Password  1-3
Pathname, setting  3-5
PE memory  5-7
Periodic software upgrades  2-1
Permissions  7-5
PMem  5-7
  partitions available  5-7

Problems
  common  7-1
    compiler logging  7-5
    compilers  7-5, 7-6
    disk space  7-5
    DISPLAY variable  7-6
    DPU access  7-2
    DPU connections  7-3
    execution  7-6
    host name  7-6
    job manager  7-3
    license manager  7-4
    MPPE  7-6
    permissions  3-8, 7-5
    remote use  3-8
    reporting  7-3, 7-4, 7-6, 7-7
    system crash  7-3
    system date  7-7
    user id  7-6

rc.local  3-2
rc.maspar  5-6
Remote DECstations  3-7
Remote setup  3-7
Reporting problems  7-3
  compiler  7-6
  execution  7-6
  job manager  7-3
  MPPE  7-7
  system crash  7-3
root file system  2-3

Selecting a monitor  2-3
server  2-13
setld command  2-4
setld command  2-6, 2-10
single-user mode  2-8, 2-12
Software subsets
  installing mandatory  2-3
  installing optional  2-3
  recommended optional subsets  2-3
Software upgrades 2-1
SPARCstations 3-8
Stand-alone environment 3-2
Standby disk 6-4, 6-8
System administration 6-1
System date 7-7
System log file 7-3

Time slice 5-8
Timing, DPU manager 5-7
tip 3-3
ttys 3-3, 3-4
Typographic conventions iii

user 7-2
uid 1-3, 3-8, 3-9, 7-6
ULTRIX 1-1, 7-2
Upgrades
installing 2-1
Upgrades, software 2-1
Upgrading
from previous releases 2-5
User id 1-3, 3-8, 3-9, 7-6
User maspar 1-3
Using the modem 3-4
/usr/maspar 3-6
Utilities 5-1

VME file system
See VMEFS
VME files
attributes of 4-3
creating 4-2
restrictions on using 4-3
supported system calls 4-3
unsupported system calls 4-3
VMEFS
file names 4-3
front-end memory mapping 4-3
mounting 4-2
nodes 4-3
number of files 4-3
restrictions 4-3
subdirectories 4-3
symbolic links 4-3
vmeaccess functions 4-2
VMEFSENDABLE 4-3

whatis 2-8, 2-11
Window manager 1-2
Windowing environment 1-2

X Window system compatibility 1-2

Yellow Pages 3-7, 3-8
updating 3-10
MasPar MP-1
Architecture Specification

Part Number 9300-5001
Revision A5, July 1991

MasPar Computer Corporation
Sunnyvale, California
MasPar MP-1 Architecture Specification
Document Part No. 9300-5001
Revision A5, July, 1991

This manual describes the MasPar MP-1 system architecture, Rev. 2.1.

MasPar Computer Corporation makes no representation or warranties regarding the contents of this document and reserves the right to revise this document or make changes in the specifications of the product described herein at any time without obligation to notify any person of such revisions or changes.

Copyright © 1990, 1991, MasPar Computer Corporation, Sunnyvale, California
All Rights Reserved
Printed in the United States of America

This manual, its format, and the hardware, software, and microcode described in it are copyrighted by MasPar Computer Corporation and may not be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the express written permission of MasPar Computer Corporation.

Restricted Rights Legend

The software and microcode documented in this manual has been developed at private expense and is not in the public domain. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subdivision (c) (I) (i) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

MasPar is a trademark of MasPar Computer Corporation.
UNIX is a registered trademark of the American Telephone and Telegraph Company.
DEC, VAX, VAXstation, and ULTRIX are trademarks of Digital Equipment Corporation.

MasPar Computer Corporation
749 North Mary Avenue
Sunnyvale, California, 94086
408-736-3300
408-736-9560 (FAX)
# Table of Contents

## Chapter 1.

### Introduction and Overview

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Overview</td>
<td>1-1</td>
</tr>
<tr>
<td>Front End (FE)</td>
<td>1-3</td>
</tr>
<tr>
<td>Array Control Unit (ACU)</td>
<td>1-3</td>
</tr>
<tr>
<td>ACU Memory Resources</td>
<td>1-4</td>
</tr>
<tr>
<td>CReg</td>
<td>1-5</td>
</tr>
<tr>
<td>CMem</td>
<td>1-5</td>
</tr>
<tr>
<td>IMem—ACU Instruction Memory</td>
<td>1-5</td>
</tr>
<tr>
<td>CSReg</td>
<td>1-5</td>
</tr>
<tr>
<td>ACU FE Registers</td>
<td>1-6</td>
</tr>
<tr>
<td>Queues</td>
<td>1-6</td>
</tr>
<tr>
<td>Processor Element Array (PE Array)</td>
<td>1-6</td>
</tr>
<tr>
<td>PE Array Programming Model</td>
<td>1-7</td>
</tr>
<tr>
<td>PE Array Communications</td>
<td>1-9</td>
</tr>
<tr>
<td>Program Execution Model</td>
<td>1-9</td>
</tr>
<tr>
<td>DPU Instruction Set Summary</td>
<td>1-10</td>
</tr>
</tbody>
</table>

## Chapter 2.

### DPU Programming Model

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operand Formats and Addressing</td>
<td>2-1</td>
</tr>
<tr>
<td>Bit and Byte Numbering</td>
<td>2-2</td>
</tr>
<tr>
<td>Floating-Point Operands and Semantics</td>
<td>2-2</td>
</tr>
<tr>
<td>VAX Floating-Point</td>
<td>2-2</td>
</tr>
<tr>
<td>IEEE Floating-Point</td>
<td>2-2</td>
</tr>
<tr>
<td>PE Numbering</td>
<td>2-3</td>
</tr>
<tr>
<td>Boundary Conditions</td>
<td>2-4</td>
</tr>
<tr>
<td>Register/Memory Model</td>
<td>2-4</td>
</tr>
<tr>
<td>ACU Register/Memory Model</td>
<td>2-4</td>
</tr>
</tbody>
</table>
## Chapter 3. The DPU Instruction Set

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPU Instruction Description Notation</td>
<td>3-2</td>
</tr>
<tr>
<td>ACU References</td>
<td>3-2</td>
</tr>
<tr>
<td>PE References</td>
<td>3-3</td>
</tr>
<tr>
<td>Addressing Modes</td>
<td>3-4</td>
</tr>
<tr>
<td>Instruction Formats</td>
<td>3-5</td>
</tr>
<tr>
<td>Instruction Specifications</td>
<td>3-7</td>
</tr>
<tr>
<td>Examples</td>
<td>3-7</td>
</tr>
<tr>
<td>Important General Rules</td>
<td>3-7</td>
</tr>
<tr>
<td>Notation</td>
<td>3-9</td>
</tr>
<tr>
<td>Condition Codes</td>
<td>3-11</td>
</tr>
<tr>
<td>ACU Move/Load/Store Instructions</td>
<td>3-12</td>
</tr>
<tr>
<td>ACU Register Move Instructions</td>
<td>3-12</td>
</tr>
<tr>
<td>ACU Register Load/Store Instructions</td>
<td>3-13</td>
</tr>
<tr>
<td>ACU Calculate Instructions</td>
<td>3-14</td>
</tr>
<tr>
<td>ACU Jump and Branch Instructions</td>
<td>3-14</td>
</tr>
<tr>
<td>ACU Unconditional Jump and Branch Instructions</td>
<td>3-15</td>
</tr>
<tr>
<td>ACU Conditional Branch Instructions</td>
<td>3-15</td>
</tr>
<tr>
<td>PE Move/Load/Store Instructions</td>
<td>3-16</td>
</tr>
<tr>
<td>PE Register Move Instructions</td>
<td>3-16</td>
</tr>
<tr>
<td>PE Special Register Move Instructions</td>
<td>3-17</td>
</tr>
<tr>
<td>PE Flag Move Instructions</td>
<td>3-17</td>
</tr>
<tr>
<td>PE Condition Code Move Instructions</td>
<td>3-17</td>
</tr>
<tr>
<td>PE Register Load/Store Instructions</td>
<td>3-18</td>
</tr>
<tr>
<td>PE Calculate Instructions</td>
<td>3-19</td>
</tr>
<tr>
<td>PE Binary Integer Instructions</td>
<td>3-20</td>
</tr>
<tr>
<td>PE Unary Integer Instructions</td>
<td>3-20</td>
</tr>
<tr>
<td>PE Integer Conversion Instructions</td>
<td>3-21</td>
</tr>
<tr>
<td>PE Boolean Instructions</td>
<td>3-22</td>
</tr>
<tr>
<td>PE Floating-Point Instructions</td>
<td>3-23</td>
</tr>
<tr>
<td>PE Floating-Point Binary Instructions</td>
<td>3-23</td>
</tr>
<tr>
<td>PE Floating-Point Unary Instructions</td>
<td>3-24</td>
</tr>
<tr>
<td>PE Floating-Point Operand Conversion Instructions</td>
<td>3-24</td>
</tr>
<tr>
<td>PE Communication Instructions</td>
<td>3-25</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Router Network</td>
<td>3-25</td>
</tr>
<tr>
<td>X-Network</td>
<td>3-26</td>
</tr>
<tr>
<td>Router Instructions</td>
<td>3-26</td>
</tr>
<tr>
<td>X-Network Instructions</td>
<td>3-28</td>
</tr>
<tr>
<td>X-Network Instruction Descriptions</td>
<td>3-28</td>
</tr>
<tr>
<td>X-Network Examples</td>
<td>3-30</td>
</tr>
<tr>
<td>PE I/O Instructions</td>
<td>3-31</td>
</tr>
<tr>
<td>Miscellaneous Instructions</td>
<td>3-32</td>
</tr>
<tr>
<td>ACU ↔ PE Move Instructions</td>
<td>3-32</td>
</tr>
<tr>
<td>Other Miscellaneous Instructions</td>
<td>3-33</td>
</tr>
<tr>
<td>graddr</td>
<td>3-33</td>
</tr>
<tr>
<td>enable</td>
<td>3-33</td>
</tr>
<tr>
<td>disable</td>
<td>3-33</td>
</tr>
<tr>
<td>Chapter 4. User/Supervisor Mode Operation</td>
<td>4-1</td>
</tr>
<tr>
<td>ACU Data Access</td>
<td>4-2</td>
</tr>
<tr>
<td>PE Data Access</td>
<td>4-3</td>
</tr>
<tr>
<td>ACU Instruction Access</td>
<td>4-3</td>
</tr>
<tr>
<td>Privileged Instructions</td>
<td>4-3</td>
</tr>
<tr>
<td>PE I/O Access</td>
<td>4-4</td>
</tr>
<tr>
<td>Changing User/Supervisor Mode</td>
<td>4-4</td>
</tr>
<tr>
<td>Chapter 5. Trap Processing</td>
<td>5-1</td>
</tr>
<tr>
<td>Trap Processing Data Structures</td>
<td>5-2</td>
</tr>
<tr>
<td>Trap Control Record</td>
<td>5-3</td>
</tr>
<tr>
<td>Trap Table</td>
<td>5-4</td>
</tr>
<tr>
<td>General Trap Programming</td>
<td>5-5</td>
</tr>
<tr>
<td>Reset Operation</td>
<td>5-7</td>
</tr>
<tr>
<td>Faults</td>
<td>5-7</td>
</tr>
<tr>
<td>System Services</td>
<td>5-9</td>
</tr>
<tr>
<td>System Debugging Support</td>
<td>5-9</td>
</tr>
<tr>
<td>Register Interrupts</td>
<td>5-10</td>
</tr>
<tr>
<td>M-Machine (M_Mach) Traps</td>
<td>5-11</td>
</tr>
<tr>
<td>Simultaneous Traps</td>
<td>5-11</td>
</tr>
<tr>
<td>Chapter 6. System Halts</td>
<td>6-1</td>
</tr>
<tr>
<td>Chapter 7. DPU Initialization (Cold Start)</td>
<td>7-1</td>
</tr>
</tbody>
</table>
Chapter 8.   ACU Interface to the FE   8-1
Appendix A.  Multiuser Operation   A-1
Index
List of Figures

Figure 1-1  System Block Diagram ................................................. 1-2
Figure 1-2  ACU Block Diagram ................................................... 1-4
Figure 1-3  PE Block Diagram ..................................................... 1-7
Figure 1-4  Synchronous Flow Control Model ............................... 1-10
Figure 2-1  PE Addressing ........................................................... 2-3
Figure 2-2  CMem Regions ......................................................... 2-20
Figure 2-3  PE Block Diagram ..................................................... 2-22
Figure 3-1  Instruction Formats ................................................... 3-6
Figure 3-2  X-Network Geography .............................................. 3-28
Figure 4-1  CMem Regions ......................................................... 4-2
vi List of Figures
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>ACU Registers: CReg</td>
<td>2-6</td>
</tr>
<tr>
<td>2-2</td>
<td>ACU Special Registers: CSReg</td>
<td>2-7</td>
</tr>
<tr>
<td>3-1</td>
<td>Addressing Mode Abbreviations</td>
<td>3-4</td>
</tr>
<tr>
<td>3-2</td>
<td>Immediate Addressing Modes</td>
<td>3-5</td>
</tr>
<tr>
<td>3-3</td>
<td>PE Flags and Meanings</td>
<td>3-8</td>
</tr>
<tr>
<td>3-4</td>
<td>PE Arithmetic Instructions</td>
<td>3-9</td>
</tr>
<tr>
<td>3-5</td>
<td>Condition Code Names and Equations</td>
<td>3-11</td>
</tr>
<tr>
<td>3-6</td>
<td>ACU Register Move Instructions</td>
<td>3-12</td>
</tr>
<tr>
<td>3-7</td>
<td>ACU Register Load/Store Instructions</td>
<td>3-13</td>
</tr>
<tr>
<td>3-8</td>
<td>ACU Calculate Instructions</td>
<td>3-14</td>
</tr>
<tr>
<td>3-9</td>
<td>ACU Unconditional Jump and Branch Instructions</td>
<td>3-15</td>
</tr>
<tr>
<td>3-10</td>
<td>ACU Conditional Jump and Branch Instruction</td>
<td>3-15</td>
</tr>
<tr>
<td>3-11</td>
<td>PE Register Move Instructions</td>
<td>3-16</td>
</tr>
<tr>
<td>3-12</td>
<td>PE Special Register Move Instructions</td>
<td>3-17</td>
</tr>
<tr>
<td>3-13</td>
<td>PE Flag Move Instructions</td>
<td>3-17</td>
</tr>
<tr>
<td>3-14</td>
<td>PE Condition Code Move Instructions</td>
<td>3-18</td>
</tr>
<tr>
<td>3-15</td>
<td>PE Register Load/Store Instructions</td>
<td>3-18</td>
</tr>
<tr>
<td>3-16</td>
<td>PE Binary Integer Instructions</td>
<td>3-20</td>
</tr>
<tr>
<td>3-17</td>
<td>PE Unary Integer Instructions</td>
<td>3-21</td>
</tr>
<tr>
<td>3-18</td>
<td>PE Integer Conversion Types</td>
<td>3-21</td>
</tr>
<tr>
<td>3-19</td>
<td>PE Integer Conversion Instructions</td>
<td>3-22</td>
</tr>
<tr>
<td>Table 3-20</td>
<td>PE Boolean Instructions</td>
<td>3-22</td>
</tr>
<tr>
<td>Table 3-21</td>
<td>PE Uncontingent Flow Control Instruction</td>
<td>3-23</td>
</tr>
<tr>
<td>Table 3-22</td>
<td>Floating-Point Binary Instructions</td>
<td>3-23</td>
</tr>
<tr>
<td>Table 3-23</td>
<td>Floating-Point Unary Instructions</td>
<td>3-24</td>
</tr>
<tr>
<td>Table 3-24</td>
<td>PE Floating-Point Conversion Types</td>
<td>3-24</td>
</tr>
<tr>
<td>Table 3-25</td>
<td>Floating-Point Conversion Instructions</td>
<td>3-25</td>
</tr>
<tr>
<td>Table 3-26</td>
<td>Router Instructions</td>
<td>3-27</td>
</tr>
<tr>
<td>Table 3-27</td>
<td>X-Network Instructions</td>
<td>3-28</td>
</tr>
<tr>
<td>Table 3-28</td>
<td>PE I/O Instructions</td>
<td>3-31</td>
</tr>
<tr>
<td>Table 3-29</td>
<td>Single-bit ACU ↔ PE Move Instructions</td>
<td>3-32</td>
</tr>
<tr>
<td>Table 3-30</td>
<td>Multi-bit ACU ↔ PE Move Instructions</td>
<td>3-32</td>
</tr>
<tr>
<td>Table 3-31</td>
<td>Miscellaneous Instructions</td>
<td>3-33</td>
</tr>
<tr>
<td>Table 5-1</td>
<td>Trap Table Assignments</td>
<td>5-4</td>
</tr>
<tr>
<td>Table 5-2</td>
<td>Simultaneous Trap Processing Priority</td>
<td>5-12</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction and Overview

The MasPar MP-1 system is a massively parallel Single Instruction, Multiple Data (SIMD) computer system with over 1,000 microprocessors, or processor elements (PEs). The system uses a register-based load/store architecture with separate instruction and data memory areas.

This manual describes the MasPar MP-1 system architecture including the Array Control Unit (ACU), the PE array, and the system instruction set. Implementation details are not described here. The *MasPar System Overview* describes the software architecture.

System Overview

The MP-1 system contains tightly integrated specialized units, providing a cost-effective, high-performance computer system.

Figure 1-1 shows a block diagram of the MasPar MP-1 system.
Figure 1-1 System Block Diagram
The MP-1 system contains the following major subsystems:

- The Front End (or FE), which is currently a UNIX subsystem
- The Data-Parallel Unit (DPU)
- The I/O Subsystem

The Front End and the I/O Subsystem are not discussed in detail in this document.

The Data Parallel Unit in turn contains the following:

- The Array Control Unit (ACU)
- The PE array
- The Interarray Communication Mechanisms, including the following:
  - The X-Network for communication with neighboring PEs
  - The router network, which enables general PE-to-PE communication
  - Two global busses:
    - A common bus on which the ACU broadcasts instructions and data to all or selected PEs
    - An OR-tree that consolidates status responses from all the PEs back to the ACU

**Front End (FE)**

The FE is a UNIX Subsystem that provides network and graphics services, including the software environment in which all the MasPar tools and utilities (such as compilers) are executed. It contains the interface to the DPU, which, through the PE array, performs the computationally intensive, parallel portions of applications.

**Array Control Unit (ACU)**

The ACU executes PE array instruction and executes instructions on the ACU proper (such as array housekeeping and FE communication). The ACU has these key characteristics:

- 32-bit, two address, load/store, RISC style instruction set
- 4-GByte virtual instruction address space, which is managed as 4096-byte pages
- Separate byte-addressed instruction and byte-addressed data spaces, both independent from the FE workstation’s virtual address space
- Both user and supervisor mode operation with protected data spaces
- Queues for interaction with the UNIX Subsystem

**ACU Memory Resources**

The ACU internal memory space includes CReg, CMem, IMem, and CSReg. Their places in the ACU are shown in Figure 1-2.

![ACU Block Diagram](image)

**Figure 1-2 ACU Block Diagram**

The following subsections describe each block. The exact boundaries are implementation-dependent.
CReg

CReg is a bank of thirty-two 32-bit registers named C0-C31. It provides the working area for all computation in the ACU.

Register C0 is special; reads always produce 0, and writes set the condition codes as if the operand were loaded, without really loading it. C0 provides a useful addressing operand for instructions; because all memory access operations use a "base plus offset" style address calculation, C0 provides a convenient source of a zero value.

CReg is the key to addressing the other memory components; many addressing modes use a value in CReg plus an offset. For details on addressing modes, see Chapter 3 of this manual.

The memory load and store instructions (cld and cst) are the only instructions that move data between CMem and CReg.

CMem

CMem is the data memory of the ACU. It is further divided into supervisor and user regions, and the supervisor region is divided further. All CMem is byte addressable, and it always matches the byte-order style of the FE (big endian or little endian).

The user region contains no inherent data structures; programs have complete freedom in using it. The supervisor region is divided into several areas, most of which contain defined data structures. These areas are described in Chapter 2 of this manual.

IMem—ACU Instruction Memory

IMem is the ACU's instruction memory; it holds all the ACU's instructions. IMem address space is separate from CMem data space, following a Harvard-style architecture. Only instruction fetches are allowed into IMem. Instructions are written into IMem from the VMEbus.

The instruction memory is managed as a demand-paged, 4-Byte virtual memory. Pages contain 4 KBytes, or 1024 32-bit words. Page faults are generated by the ACU and handled by the FE.

CSReg

The thirty-two 32-bit registers in CSReg, such as the Program Status Word (PSW), provide system status information and control of the ACU. These are described in detail in Chapter 2 of this manual.
ACU FE Registers

The FE interacts with the ACU by reading and writing special interface registers in the ACU. These registers are memory-mapped into the FE address space.

Queues

The FE Interface registers of the ACU provide, among other things, a communication path between the FE and the ACU. The To Back End Queue (TOBEQ) carries data from the FE to the ACU, and the From Back End Queue (FRBEQ) carries data from the ACU to the FE.

Both queues buffer 32-bit words on a first-in, first-out basis in their respective directions. During normal operation, when the ACU is running, the instruction \texttt{cmov.Cr.Fb} moves a word from CReg to the FRBEQ, and the instruction \texttt{cmov.Tb.Cr} moves a word from the TOBEQ to CReg.

The FE can add an entry to the TOBEQ or remove one from the FRBEQ by accessing the ACU FE Interface registers.

Processor Element Array (PE Array)

The PE array can consist of thousands of PEs per system. All PEs work in parallel to execute a common instruction stream, but they each have their own individual data. This is the basic principle of SIMD operation. Individual PEs can either execute or ignore an instruction based on a data-dependent condition code.

The programmer very rarely has to program individual PEs. Almost all of the PE array instructions and the registers in the ACU control space deal with the entire array. Only a few specialized instructions allow the user to access a single PE, and these are primarily used for diagnostics and debuggers. The closest a programmer comes to programming individual PEs is to use the condition codes to enable or disable PEs, thus dividing the PEs into two groups—those for which the condition is true, and those for which it is false.

Each PE contains a private register set (PReg) and a private memory bank (PMem). Register operations, which are much faster than RAM operations, can be performed on 1-, 8-, 16-, 32-, and 64-bit values. Operations on operands other than one-bit operands need to be byte-aligned.

Figure 1-3 shows the block structure of a PE.
Figure 1-3 PE Block Diagram

**PE Array Programming Model**

A programmer sees the following DPU characteristics:

- **Separate instruction and data spaces.** Because this is the SIMD part of the system, a single instruction store, called ACU instruction memory, provides the instruction stream for the ACU and all the PEs.

- **Thousands of PEs, each with local PE registers and memory.** The registers can be bit-addressed or used as variable-length registers.

PReg access is roughly 10 times faster than PMem access. The ACU logic design allows the execution of PMem loads and stores to PMem concurrently with nonconflicting PReg operations. If there is a conflict, the hardware stalls the PReg operation until the PMem operation is done. Attempts to address nonexistent PMem result in machine exceptions.

PMem is only directly accessible to the PE that it belongs to. However, a PE can access another PE’s memory by sending a message to the other PE and requesting that it send the desired memory item. This method is slower than a PMem or PReg access.

- **Data types that use the same bit layout as the FE.**

- **Register-to-register operations that can be performed on data items of 1, 8, 16, 32, or 64 bits.**
• Load and store operations that move data between PMem and PReg. These load and store instructions execute concurrently with PE ALU register-to-register operations. The address can be "direct" or "indirect". In a direct address, the ACU broadcasts the same address from a CReg to all PEs. In an indirect address, each PE gets its PMem address from the same location in its PReg, so each PE can access a different address. Indirect address operations take significantly longer than direct address operations.

There are two types of operations: memory operations (loads and stores) and execution operations (all others). Memory operations run concurrently with execute operations. If there is a conflict where both need to use the same PReg, the execute operations stall until the memory operations are complete.

• Support for floating-point operations. The supported format is the same as that of the PE. A register mode bit allows a program to select either of the VAX 64-bit formats (D or G); IEEE format is also supported.

• Properties specific to SIMD:

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared instruction stream</td>
<td>The PEs all share a single instruction stream.</td>
</tr>
<tr>
<td></td>
<td>Because of this, each PE knows what the others are doing, which leads</td>
</tr>
<tr>
<td></td>
<td>to a specific programming style. For example, queues and semaphores</td>
</tr>
<tr>
<td></td>
<td>are not required.</td>
</tr>
<tr>
<td>Execution contingent bit</td>
<td>This mechanism is a critical part of practical SIMD programming.</td>
</tr>
<tr>
<td></td>
<td>On any given PE instruction, each PE is either active or inactive,</td>
</tr>
<tr>
<td></td>
<td>depending on the state of its E-bit, which determines whether or not</td>
</tr>
<tr>
<td></td>
<td>the PE responds to the instruction stream broadcast by the ACU.</td>
</tr>
<tr>
<td>Memory contingent bit</td>
<td>Because the ACU runs memory operations concurrently with execute</td>
</tr>
<tr>
<td></td>
<td>operations, each PE has a second contingent bit called the M-bit.</td>
</tr>
<tr>
<td></td>
<td>The M-bit determines whether or not that PE participates in memory</td>
</tr>
<tr>
<td></td>
<td>(PReg ↔ PMem) operations.</td>
</tr>
</tbody>
</table>

The way the PReg is addressed helps with virtual programming. Because PRegs are addressed as \( CReg + offset \), once a sequence of instructions is constructed with the same CReg but differing offsets, all the programmer has to do is change the value of CReg, and the same code sequence can be used over again in another part of PReg. Because loads (PMEM to PReg) and stores (PReg to PMem) overlap with other PE operations, a PE can perform calculations during memory access.

Each PE has a specific address consisting of its row and column address. Visually speaking, PE (0,0) is located in the upper left-hand corner, with rows increasing downwards, and columns increasing towards the right. By convention, the row appears first, and the column appears second, and both are numbered starting from 0. Therefore, PE (3,2) is (starting from the upper left-hand corner) four down and three to the right. The number of rows and columns changes according to the number of PE array boards installed.
PE Array Communications

The PE array supports these four communications mechanisms:

- The **X-Network**, or X-Net, allows selected PEs to move data to a neighboring PE, one bit per clock cycle (after a startup overhead). Any number of PEs can take part in an X-Net operation; however, all “sending” PEs must send data from the same internal (to the PE) address to a neighbor in the same relative geographical position. For example, all selected PEs can send data from the same internal address to the neighbor to the west.

- The **global router network** transmits data in bit-serial format, one bit per clock cycle, from one PE to any other PE. Router operations are distance-insensitive. This enables the accessing PE to have global access to all the data in the PE array. Point-to-point (router) communications are the only means by which any kind of global data reference occurs in the PE array.

- The **broadcast network** allows data to be broadcast from the ACU to all PEs. Each PE can independently elect to receive the data.

- The **OR-Reduction network** provides a boolean logical OR reduction network that enables global PE data conditions to be easily detected.

Program Execution Model

How users see the MP-1 depends on how they are interacting with it:

- Using high-level languages, one source drives all parts and the compiler manages interactions. These languages automatically generate two cooperating pieces of code: one for the FE and one for the DPU.

- Using a lower-level language, there is one source for the DPU and one source for the FE. The programmer explicitly manages the DPU-FE interactions by means of subroutine calls that communicate between the two source programs.

Figure 1-4 shows a synchronous programming model for the MP-1. This is the simplest programming model for the FE and DPU running cooperating code. The MP-1 supports both synchronous and asynchronous control models.

Another way to look at the control flow is to analyze what happens during the life of a high-level program:

1. The programmer codes a program in one of the MasPar languages, then compiles it.

2. The compiler generates two blocks of code: one for the FE and one for the DPU. The code in the FE executes the inherently serial part of the user calculations. Running the DPU code, the ACU performs serial housekeeping related to PE array calculations and controls execution of parallel calculations by the PE array.
Figure 1-4 Synchronous Flow Control Model

3. While the program is executing, the FE communicates with the ACU through the queues and DMA.

4. The ACU generates microinstructions to control the PE array and I/O devices.

DPU Instruction Set Summary

The DPU instruction set is composed of one, two, or three 32-bit word instructions. Even though all instructions are stored in the ACU, the instructions affect either the ACU or the PE array. To differentiate the two types of instructions, ACU instructions always start with c (for aCu) or b (for Branch). PE instructions never start with these letters. The instructions are further separated into the following basic groups:
ACU jump and branch

The ACU Jump and Branch provide a mechanism to change instruction sequencing by directly modifying the program counter. Instructions for conditional branches (beq, bne, etc.), jumps (cjmp), and subroutine calls are provided.

ACU calculate

The ACU provides direct computational capability using a two-operand style instruction where all operations are register to register. Operations for add (cadd), subtract (csub), and (cand), or (cor), etc. are provided. All operations manipulate 32-bit operands.

ACU load/store

Only load/store instructions are allowed between the register set and ACU data memory (cl/cst). Base plus offset style address modes allow flexibility in generating memory references.

PE calculate

PE calculate instructions are similar to ACU calculate instructions in that they are also two-operand register to register operations. Integer operations like add, mul, div, and, xor, etc. are provided for 1-, 8-, 16-, 32-, and 64-bit operands. Floating-point operations are provided for both 32- and 64-bit operands. Floating-point operations include fadd, fdiv, fmul, and frem.

PE load/store

Operations between PE registers (PReg) and PE Memory (PMem) are also provided for 8-, 16-, 32-, and 64-bit operands, in both addressing modes:

- Direct: where all the PE access the same location (ld/st)
- Indirect: where each PE accesses a unique location (ldx/stx)

PE communications

PE communication instructions provide direct access to both X-Net and global router operation. For the X-Net instructions, the distance and direction are specified. Router operation is partitioned into four basic operations:

- ropen: open a connection
- rsend: send data from the originating PE to the destination PE
- rfetch: get data from the destination PE back to the originator
- rclose: close the connection

Misc

The miscellaneous instructions provide a variety of functions including:

- Instructions to control which PEs are enabled.
- No operation (nop)
- System calls (cscall)
• Breakpoints(cbreak)

• Instructions for moving data between the special registers (csmov: CReg ↔ CReg and smov: SReg ↔ PReg)

• Instructions to move data between the ACU and PE array (gor, dist)

Chapter 3 of this manual provides details on the DPU instruction set.
Chapter 2

DPU Programming Model

The DPU consists of the ACU and the PE array.

This chapter describes the DPU programming model in terms of the following:
- Operand formats and addressing for ACU and PE operations
- The ACU register and memory model
- The PE register and memory model

Operand Formats and Addressing

The DPU uses the following operand formats:
- ACU operations: 32-bit (word) integer operands
- PE operations:
  - Bits
  - Integers: 8-bit bytes, 16-bit half-words, 32-bit words and 64-bit double words
- Floating-Point Operands:
  - 32-bit single-precision
  - 64-bit double-precision

The floating point format (VAX or IEEE) is determined by the FE workstation and is a fixed microcode option.
Bit and Byte Numbering

For all other operands, bit numbering starts with zero at the least significant bit (LSB). For example, the following shows the representation for an 8- and 16-bit operand:

8-bit operand:

\[
\text{MSB} \rightarrow \begin{array}{cccccc}
7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
\end{array} \quad \text{LSB}
\]

16-bit operand:

\[
\text{MSB} \rightarrow \begin{array}{cccccccccccccc}
15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
\end{array} \quad \text{LSB}
\]

Byte numbering can be done in either big-endian or little-endian fashion. In all cases, the MasPar system uses the same style as the front-end workstation. The VAX format uses the little-endian style, with increasing byte addresses containing increasingly significant operand bits. For example, the bits of a 16-bit VAX integer are stored as follows, with the top row representing Byte 0 and the bottom row representing Byte 1:

<table>
<thead>
<tr>
<th>Addr: n</th>
<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>Addr: n+1</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

Floating-Point Operands and Semantics

The PE array can perform either IEEE- or VAX-format floating point. However, it can only operate in one mode at a time; this is always the same floating point system used by the FE. (That is, PEs associated with a VAX FE use VAX FP, while PEs associated with an FE that uses IEEE FP also use IEEE FP.)

**VAX Floating-Point**

PE operations support F, D, and G VAX formats for single (32-bit) and double (64-bit) precision floating point representations. The specifics of the supported formats follow:

- **F**: 23-bit mantissa, 8-bit exponent, 32-bit
- **D**: 55-bit mantissa, 8-bit exponent, 64-bit
- **G**: 52-bit mantissa, 11-bit exponent, 64-bit

The semantics of VAX mode PE floating point operations are defined as "identical to a VAX".

**IEEE Floating-Point**

PE operations support regular IEEE single precision (32-bit) and double precision (64-bit) formats. The DPU does not support IEEE extended precision formats.
The specifics of the supported formats follow:

- Single: 23-bit mantissa, 8-bit exponent, 32-bit
- Double: 52-bit mantissa, 11-bit exponent, 64-bit

IEEE mode floating point operations in the PEs conform to all required aspects of the ANSI/IEEE 754-1985 Standard for Binary Floating Point Arithmetic, with the following exceptions.

Round to Nearest Mode
If the two representable values nearest the exact result are equally near, the one with larger absolute value is generated. IEEE-754 specifies that the one with zero in its least significant bit be generated.

Other Rounding Modes
Only one rounding mode, Round to Nearest, is supported.

Denormalized Numbers
A denormalized number appearing as an operand is always treated as if it were zero. Where IEEE-754 specifies a denormalized result, a zero is generated instead.

Inexact Exception
The inexact exception is never signaled.

IEEE-754 requires that both signaling and quiet NaNs be supported, but does not specify how their values are distinguished. IEEE mode floating point operations in the PEs use the most significant bit (MSB) of the stored fraction (not the hidden bit) to distinguish between signaling and quiet NaN values. A NaN with a 1 in the fraction’s MSB is a signaling NaN, while a NaN with a 0 in the fraction’s MSB is a quiet NaN.

**PE Numbering**

The system allows for a variable number of processors in the system where the physical layout is in a two-dimensional grid. PEs are addressed in a configuration-specific manner in a row-column format, as shown in Figure 2-1.

- The origin of the grid is at the top left corner.
- The processor at the top left corner of the grid is processor (0,0).
- The number of columns always either equals the number of rows (a square array) or is twice the number of rows (a rectangular array).
- The numbers of both rows and columns are a power of two.

![PE Addressing](image-url)

**Figure 2-1 PE Addressing**
The size of the row and column are determined by the machine configuration. For example, a machine with 16,384 processors has both 128 rows and 128 columns, which means that the width of both the row and column field is seven. The row and column are concatenated so that PE column addresses can be incremented and overflow into the row address for a linear view of the PE array.

A cluster is a group of adjacent PEs that share a single memory system and router channel. The cluster boundaries are fixed in hardware. PEs within a cluster differ only in the low-order bits of their row and column addresses.

- If you know that only a single PE in any cluster is involved in a load or store, the load or store solitary instruction (ldsol or stdsol) is more efficient than a normal load or store. These instructions are described in Chapter 3.
- Knowing cluster boundaries can be important when planning the use of the router, as described in Chapter 3.

**Boundary Conditions**

For the purposes of the X-Net communication operations, the array edges are interconnected in a toroidal manner: the topmost row of PEs is connected to the bottom-most row, and the rightmost PEs are connected to the leftmost PEs. For example, data transferred “north” out of the top is received from the “south” at the bottom. This interconnection is known as a “toroidal wrap”

**Register/Memory Model**

This section discusses each ACU and PE register and memory area in detail. Figure 1-1 above shows the system block diagram.

**ACU Register/Memory Model**

The ACU has the following six logical register/memory blocks, shown in Figure 1-2 and mentioned in Chapter 1:

- **IMem**: The ACU instruction memory. It is virtually addressed, up to 4 GBytes. Instruction memory cannot be changed by a DPU program.
- **CReg**: Thirty-two 32-bit general-purpose registers (C0-C31). Register C0 always returns a zero value.
- **CSReg**: Thirty-two 32-bit special registers (CS0-CS31). They provide access to special processor status information, including the ACU processor status word (CPSW) in CS0.
- **PC**: The 32-bit program counter. It points to the currently executing instruction.
CMem Memory containing user data, microcode data, and supervisor data. Also, sections of the VME bus are directly mapped into the CMem address space. Different accesses are allowed, depending on user/supervisor state.

FE-DPU Interface Four 4-KByte pages, memory-mapped onto the VME bus so that the FE can access DPU resources. Specific memory mapped registers provide DPU command, control, and data access.

The following sections describe in more detail each of the ACU register/memory blocks.

**Register/Memory Access Table Conventions**

Each of the Register/Memory tables uses a similar format. In each table, the legal access modes are specified and have the following meaning:

- **R** Read-only access. Writes are ignored.
- **W** Write-only access. Reads return zero.
- **R/W** Read and Write access.
- **RC** Read Clear. Reads return the value. For a write operation, the resulting value is determined on a bit-by-bit basis. Writes of a 0 do not effect the corresponding bit value. Writes of a 1 clear the corresponding bit value.
- **0** Reads always return a 0 and writes are ignored.

The following legal access mode modifiers are used in the tables:

- **S** Supervisor-mode access only.
- **U** User-mode access: user and supervisor-mode access allowed.
- **U*** User special access: User-mode read access allowed; user mode write access differs bit by bit.

For example, the notation S-R/W means that read/write access is only allowed in supervisor mode. Illegal accesses for memory operations can cause a memory limit trap. For special move instructions (CSMOV and SMOV), illegal accesses on reads return zero, and writes are ignored. For some registers this is on a bit-by-bit basis.

**IMem—ACU Instruction Memory**

All ACU instructions are held in IMem, the ACU instruction memory. Further, IMem address space is separate from ACU data space, following a Harvard-style architecture. Only instruction fetch accesses are allowed into IMem. Instructions are written into IMem using a DMA port from the VMEbus.

IMem is organized as a virtually addressed 4-GByte address space using 4-KByte pages. When the program counter encounters page faults, the ACU instruction fetch unit stalls after signalling a page fault interrupt on the VME bus. After the requested page is fetched by the FE, ACU execution continues.
CReg—ACU General Purpose Registers

The thirty-two 32-bit general-purpose registers, known as CReg, provide the working area for all computation in the ACU. The only operations that access memory are load and store (cld and cst). The registers are named C0-C31. Register C0 is special: it always reads zero (0), and writes have no effect except to set the condition code bits based on what the stored value would have been. C0 can be used as an addressing operand for instructions. All memory access operations use a "base plus offset" style address calculation; therefore, C0 provides a convenient source of a zero base value. C31 is also special, since it is used by the subrouting call instructions such as bsr and cjsr instructions to store the return PC location. All other registers are completely general purpose. Table 2-1 shows the CReg layout.

<table>
<thead>
<tr>
<th>Name</th>
<th>Access</th>
<th>Description</th>
<th>Init. Val.</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>U-R/W</td>
<td>Read 0; writes set ALU status</td>
<td>0</td>
</tr>
<tr>
<td>C1-30</td>
<td>U-R/W</td>
<td>General purpose registers</td>
<td>0</td>
</tr>
<tr>
<td>C31</td>
<td>U-R/W</td>
<td>Used by cjsr and bsr for return PC</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2-1  ACU Registers: CReg

CSReg—ACU Special Registers

These thirty-two 32-bit registers provide system status information and control data used by the ACU. They are accessed through CS0 through CS31.

Some of these registers have values maintained directly by the ACU hardware. For example, CS0 is the ACU program status word (CPSW); when the ACU executes an add instruction (cadd), certain bit fields (condition codes) in the CPSW are automatically updated. Typically, a user program does not have to access any of the CSReg values (though it does have access to the condition codes in the CPSW). Write access to CSRegs is implementation-dependent and can be restricted to supervisor-mode only.

Table 2-2 defines CSReg. Some of these registers are described only as Impl. dep. (implementation dependent). The table is followed by detailed definitions of the registers not so described.
<table>
<thead>
<tr>
<th>Num.</th>
<th>Name</th>
<th>Access</th>
<th>Description</th>
<th>Init. Val.</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS0</td>
<td>CPSW</td>
<td>U*-R/W</td>
<td>ACU PSW</td>
<td>0x80</td>
<td>2-8</td>
</tr>
<tr>
<td>CS1</td>
<td>QSR</td>
<td>U-R</td>
<td>Queue Status Register</td>
<td>0</td>
<td>2-9</td>
</tr>
<tr>
<td>CS2</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS3</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS4</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS5</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS6</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS7</td>
<td>CMEMLIM</td>
<td>U-R, S-R/W</td>
<td>CMem limit value</td>
<td>undefined</td>
<td>2-9</td>
</tr>
<tr>
<td>CS8</td>
<td>PMEMLIM</td>
<td>U-R, S-R/W</td>
<td>PMem limit value</td>
<td>undefined</td>
<td>2-9</td>
</tr>
<tr>
<td>CS9</td>
<td>PREGLIM</td>
<td>U-R, S-R/W</td>
<td>PReg limit value</td>
<td>undefined</td>
<td>2-10</td>
</tr>
<tr>
<td>CS10</td>
<td>PMSTAT</td>
<td>U-R, S-R/W</td>
<td>PMem status code</td>
<td>0</td>
<td>2-11</td>
</tr>
<tr>
<td>CS11</td>
<td>TRAPVAL</td>
<td>U-R, S-R/W</td>
<td>Trap parameter value</td>
<td>0</td>
<td>2-11</td>
</tr>
<tr>
<td>CS12</td>
<td>FLTCOD</td>
<td>U-R, S-R/W</td>
<td>Fault status code</td>
<td>0</td>
<td>2-12</td>
</tr>
<tr>
<td>CS13</td>
<td>FLTMSK</td>
<td>U-R, S-R/W</td>
<td>Fault mask</td>
<td>0xFF</td>
<td>2-17</td>
</tr>
<tr>
<td>CS14</td>
<td>TRAPREQ</td>
<td>U-R</td>
<td>Trap request</td>
<td>0</td>
<td>2-18</td>
</tr>
<tr>
<td>CS15</td>
<td>TRAPMSK</td>
<td>U-R, S-R/W</td>
<td>Trap mask</td>
<td>0xFF</td>
<td>2-18</td>
</tr>
<tr>
<td>CS16</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS17</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS18</td>
<td>IOINDEX</td>
<td>U-R/W</td>
<td>Index into I/O table</td>
<td>0</td>
<td>2-18</td>
</tr>
<tr>
<td>CS19</td>
<td>IOTBLBASE</td>
<td>U-R, S-R/W</td>
<td>I/O table base address</td>
<td>0</td>
<td>2-19</td>
</tr>
<tr>
<td>CS20</td>
<td>IOTBLSZ</td>
<td>U-R, S-R/W</td>
<td>I/O Table size</td>
<td>0</td>
<td>2-19</td>
</tr>
<tr>
<td>CS21</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS22</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS23</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS24</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS25</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS26</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS27</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS28</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS29</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CS30</td>
<td>SWAP</td>
<td>U-R, S-R/W</td>
<td>Swap on trap</td>
<td>0</td>
<td>2-19</td>
</tr>
<tr>
<td>CS31</td>
<td>-</td>
<td>-</td>
<td>Impl. dep.</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2-2  ACU Special Registers: CSReg
This holds both condition flags and ACU modes.

**CC**  ACU C Condition Flag (carry)  Bit 0—[U-R/W]
CV  ACU V Condition Flag (overflow)  Bit 1—[U-R/W]
CZ  ACU Z Condition Flag (zero)  Bit 2—[U-R/W]
CN  ACU N Condition Flag (neg.)  Bit 3—[U-R/W]
CF  VAX 64-bit FP Mode  Bit 4—[U-R/W]

1 means VAX G mode.
0 means VAX D mode.

**Note:** In IEEE mode, this bit is reserved.

**US**  User/Supervisor Mode  Bit 5—[U-R, S-R/W]

Only writable in supervisor mode.

**SS**  Single Step Mode  Bit 6—[U-R, S-R/W]

1 means "trap after one instruction"
0 means "normal operation"

Only writable in supervisor mode.

**ID**  Interrupt Disable  Bit 7—[U-R, S-R/W]

1 means "no maskable traps"
0 means "allow maskable traps"

Only writable in supervisor mode.
The Queue status register contains the two status bits of the front-end interface queues.

- FRBEQFULL is 1 when the FRBEQ is full; else 0.
- TOBEQDATA is 1 when the TOBEQ contains data; else 0.

The value in this register is the lowest CMem address that causes a CMEMLIMIT violation if used by a user-mode process. The limit is a byte address that must be word-aligned, so the two LSBs must be zero. (The two LSBs are ignored in limit checking; their value is undefined if a nonzero value is stored into them.) The limit must be neither zero nor greater than CMEMSZ. (The limit is undefined if zero or a value greater than CMEMSZ is stored into it.) No limit checking is performed in supervisor mode.

The value in this register is the lowest PMem address that causes a PMEMLIMIT violation if used by a user-mode process. The limit is a byte address that must be double-word aligned, so the three LSBs must be zero. (The three LSBs are ignored in limit checking; their value is undefined if a nonzero value is stored into them.) The limit must be neither zero nor greater than PMEMSZ. (The limit is undefined if zero or a value greater than PMEMSZ is stored into it.) No limit checking is performed in supervisor mode.
The value in this register is the lowest PReg address that causes a PREGLIMIT violation if used by a user-mode process. The limit is a byte address that must be double-word aligned, so the three LSBs must be zero. (The three LSBs are ignored in limit checking; their value is undefined if a nonzero value is stored into them.) The limit must be neither zero nor greater than PREGSZ. (The limit is undefined if zero or a value greater than PREGSZ is stored into it.) No limit checking is performed in supervisor mode.
PMSTAT       PMem Status Word. CS10 [U-R, S-R/W]

28
1 1 1 1
31
4 3 2 1 0

UPMEMECC  SPMEMECC  TPMEMECC  SOLERR  zero

Each nonzero bit represents an error indication. Each individual bit is set only when its error occurs. When accessing this word, all pending PMem operations are first completed. The error conditions are only cleared by an explicit cmov instruction to the word. SPMEMECC is a status bit only and does not generate a trap request. The following describes the bit definitions:

UPMEMECC  Uncorrectable PMem ECC Bit 0
           Imprecise M_Mach Trap
This bit is set when a PMem reference results in an uncorrectable (hard) ECC error in one or more PEs. Those PEs each set their ECC_HARD bit in their processor status word (PPSW).

SPMEMECC  Soft PMem ECC Bit 1
           This bit is set when a PMem reference results in a correctable (soft) ECC error in one or more PEs. Those PEs each set their ECC_SOFT bit in the PPSW.

TPMEMECC  Too many soft PMem ECC Bit 2
           Imprecise M_Mach Trap
This bit is set when a PMem reference results in a soft ECC error, where PMEMECC (the PMem soft ECC count in supervisor CMem) exceeds five corrected errors.

SOLERR    Solitary load/store error Bit 3
           Imprecise M_Mach Trap
This bit is set when a "solitary" load or store instruction is executed and more than one involved PE in a cluster has its M-bit set.

TRAPVAL    Trap Parameter Value CS11 [U-R, S-R/W]

32
31

This is set by the ccall and cbreak instructions.
Be aware of the following aspects of FLTCOD:

- The entire word is written by the hardware each time a fault trap occurs, and at no other time; each nonzero bit represents an enabled fault which is causing the trap. In other words, the hardware changes the value of FLTCOD and takes a fault trap when (and only when) one or more faults occur which are enabled by their corresponding FLTMSK bits; the new FLTCOD value has a one for each such fault, and zeros for all other faults. The effect is that FLTCOD, unless it has been explicitly modified by a csmov instruction since the most recent fault trap, always has ones in exactly those bits which caused the most recent fault trap.

- Supervisor writes do not cause a fault trap. User mode writes have no effect.
• No memory limit violations can be generated in supervisor mode. Therefore, faults CMEMLIMIT, PREGLIMIT, and PMEMLIMIT can only occur in user mode.

The following bit definitions define the conditions causing each fault, and specify whether the fault is precise. Note that the phrase "this bit is set when ..." in these definitions actually means "this bit is set when it is enabled by its corresponding FLTMSK bit and when ...".

**CTLACUPARITY** Control store ACU parity Bit 0
Imprecise Fault
This bit is set when a parity error is detected in the ACU-control bits of a microcode instruction during the fetch from control store or the distribution of these bits.

**CTLPEPARITY** Control store PE parity Bit 2
Imprecise Fault
This bit is set when a parity error is detected in the PE-control bits of a microcode instruction, during the fetch from control store or the distribution of these bits.

**CTLIOPARITY** Control store I/O parity Bit 3
Imprecise Fault
This bit is set when a parity error is detected in the I/O-control bits of a microcode instruction, during the fetch from control store or the distribution of these bits.

**CMEMLIMIT** CMem limit Bit 4
Imprecise Fault
This bit is set when a CMem address equals or exceeds CMEMLIMIT. This fault can only occur in user mode.

**PREGLIMIT** PReg limit Bit 5
Imprecise Fault
This bit is set when a PReg address equals or exceeds PREGLIMIT. This fault can only occur in user mode.

A PREGLIMIT fault can corrupt bits in the PPSW unpredictably.

**PMEMLIMIT** PMem Limit Bit 6
Imprecise Fault
This bit is set when a PMem address equals or exceeds PMEMLIMIT. This fault can only occur in user mode.

Depending on the addressing modes of the instruction causing the fault, the offending PMem address might have been specified by the ACU or
by one or more PEs. Because this is an imprecise
fault, there is in general no way to identify the
offending instruction or its addressing modes.

If the PMem address was specified by the
individual PEs, those PEs that detected an illegal
address also set their PMEMLIMIT bit in the
PPSW. However, because this is an imprecise
fault, an instruction executed after the offending
instruction, but before the fault occurs, can clear
or otherwise modify the PMEMLIMIT bit in the
PPSW before the fault handler can examine it.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMEMPARITY</td>
<td>Instruction memory parity</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Imprecise Fault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set when the ACU program memory suffers a parity error.</td>
<td></td>
</tr>
<tr>
<td>RTRPARITY</td>
<td>Router parity</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Imprecise Fault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set when one or more PEs detects a Router communication parity error. Those PEs each set their RTR_ERR bit in the PPSW. However, because this is an imprecise fault, an instruction executed after the offending instruction but before the fault occurs can clear or otherwise modify the RTR_ERR bit in the PPSW before the fault handler can examine it.</td>
<td></td>
</tr>
<tr>
<td>XNETPARITY</td>
<td>X-Net parity</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Imprecise Fault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set when one or more PEs detects an X-Net communication parity error. Those PEs each set their XNET_ERR bit in the PPSW. However, because this is an imprecise fault, an instruction executed after the offending instruction but before the fault occurs can clear or otherwise modify the XNET_ERR bit in the PPSW before the fault handler can examine it.</td>
<td></td>
</tr>
<tr>
<td>RTRTNOR</td>
<td>Router T no R</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Precise Fault</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set when an rclose or rfetchchc instruction, when it is about to close a router connection, finds that the PPSW T flag is true in one or more PEs but the PPSW R flag is false in all PEs. There are two underlying causes for this situation.</td>
<td></td>
</tr>
</tbody>
</table>

1. A router parity error during open or send can prevent the R flag from being set in some PE where it should be set, without being detected during the router open
operation. (For example, a stuck-at-zero router wire can zero the data sent to open a connection, causing a connection request to be lost enroute.)

2. An incorrect MPAS program can explicitly modify the T or R flag in a way that causes this situation. The T and R flags are normally manipulated solely by router instructions. The only safe direct manipulations of the T and R flags are those that do not change the number of true T flags or R flags in any PE cluster; these safe manipulations do not cause a later RTRTNOR error.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Bit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREGALIGN</td>
<td>PReg alignment</td>
<td>12</td>
<td>Imprecise Fault</td>
</tr>
<tr>
<td>IMEMALIGN</td>
<td>IMem alignment</td>
<td>13</td>
<td>Imprecise Fault</td>
</tr>
<tr>
<td>CMEMALIGN</td>
<td>CMem alignment</td>
<td>14</td>
<td>Imprecise Fault</td>
</tr>
<tr>
<td>ILLOP</td>
<td>Illegal operation</td>
<td>16</td>
<td>Precise Fault</td>
</tr>
<tr>
<td>MACHFAULT</td>
<td>Machine fault</td>
<td>20</td>
<td>Imprecise Fault</td>
</tr>
<tr>
<td>IOERR</td>
<td>I/O error</td>
<td>21</td>
<td>Precise Fault</td>
</tr>
</tbody>
</table>

This bit is set when a PReg address for a non-bit macroinstruction is not byte-aligned (that is, when the three LSBs of the address are not zero).

This bit is set when the target of a branch or jump operation is not word-aligned (that is, when the two LSBs of the address are not zero).

This bit is set when a load or store address to CMem is not word-aligned (that is, when the two LSBs of the address are not zero).

This bit is set when an unspecified instruction is encountered, or when a supervisor-mode-only instruction is attempted in user mode.

This bit is set when a fatal machine error occurs (such as the microcode detecting an inconsistent machine state). This indicates a hardware malfunction.

This bit is set when an error is encountered during an I/O instruction.
If the error is due to a hardware problem, another FLTCOD bit, such as RTRPARTITY or BUSERR, is set at the same time to indicate the nature of the error. If no such bit is set at that time, the error indicates a protection violation by the I/O instruction.

**RPROTO**

Router protocol error

Bit 22

Precise Fault

This bit is set when a protocol error is encountered during a global router operation.

The possible causes of protocol errors are:

1. Attempting to open a router connection when one is already open
2. Attempting to close a router connection when no connection is open
3. Attempting to use a router connection (with rsend or rfetch) when no connection is open
4. Attempting to perform a router I/O operation (with iord or iowrt) when a router connection is open

For example, executing an rsend before executing an ropen would result in an RPROTO error.

**ILLPEADDR**

Illegal PE address

Bit 23

Precise Fault

This bit is set when an illegal PE address is encountered. A PE address, or PE number, uniquely identifies a particular PE. For example, an enable instruction with an argument specifying a PE that does not exist in the current configuration causes an ILLPEADDR error.

**BUSERR**

Bus error

Bit 19

Precise Fault

This bit is set when a VME bus error occurs.

**DIV0**

Integer divide by zero

Bit 24

Precise Fault

This bit is set when either the ACU or one or more PEs attempts a div by zero or a mod by zero. The fault trap handler must examine the opcode of the trapping instruction to determine whether it was an ACU or PE instruction. If it was a PE instruction, those PEs that attempted to
### DPU Programming Model

Divide by zero can be identified by examining their C (carry) status flag.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINEXACT</td>
<td>Floating-point inexact result</td>
<td>27</td>
</tr>
<tr>
<td>FINVOP</td>
<td>Floating-point invalid operand</td>
<td>28</td>
</tr>
<tr>
<td>FDIV0</td>
<td>Floating-point divide by zero</td>
<td>29</td>
</tr>
<tr>
<td>FUNFLO</td>
<td>Floating-point underflow</td>
<td>30</td>
</tr>
<tr>
<td>FOVFLO</td>
<td>Floating-point overflow</td>
<td>31</td>
</tr>
</tbody>
</table>

This bit is set when any PE stores a one into PPSW<FX>, regardless of whether PPSW<FX> was already one. Note that because PPSW<FX> is never set, FINEXACT is never set; therefore, this bit is effectively reserved.

This bit is set when any PE stores a one into PPSW<FI>, regardless of whether PPSW<FI> was already one.

This bit is set when any PE stores a one into PPSW<FD>, regardless of whether PPSW<FD> was already one.

This bit is set when any PE stores a one into PPSW<FU>, regardless of whether PPSW<FU> was already one.

This bit is set when any PE stores a one into PPSW<FV>, regardless of whether PPSW<FV> was already one.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLTMSK</td>
<td>Software Fault Mask</td>
</tr>
</tbody>
</table>

FLTMSK bit definitions have a one-to-one correspondence with the fault code word, FLTCOD. A fault is allowed only if its corresponding FLTMSK bit is 1. Read-only access in user mode.
This contains bits that correspond to each of the eight maskable trap requests as shown in the Trap Table, Table 5-1, page 5-4.

Each bit is set when the event occurs which corresponds to its trap. Currently, only five of the eight entries are used: CTRL0A-CTRL0D, and M_Mach. The interrupt registers, CTRL0A-CTRL0D, set the appropriate request bit each time a VME write occurs to the interrupt register. The M_Mach request bit is set each time one of the PMSTAT bits which causes an M_Mach trap is set. Each bit is cleared when the interrupt that it requests is serviced by taking a trap to the appropriate handler. Read-only access in both user and supervisor modes.

The bit definitions of the trap mask word have a one-to-one correspondence with the trap request word. A trap is allowed only when its corresponding TRAPMSK bit is one. Read-only access in user mode.

As part of the execution of every I/O instruction, the value in this register is added to the value of IOTBLBASE. This gives the address in CMem of the I/O table entry that the instruction must use. Because I/O table entries are four words long and fully aligned, the four low-order bits of IOINDEX must be 0; if they are not, they are ignored and treated as if 0. The value of IOINDEX must be less than that of IOTBSLSZ, otherwise the execution of the I/O instruction causes an IOERR fault.
IOTBLBASE  I/O Table Base Address  CS19 [U-R, S-R/W]

The value in this register is the address in CMem of the I/O Table. IOINDEX is added to this value to obtain the address of an entry in the I/O Table for use with an I/O instruction. The I/O Table is normally maintained by the ACU kernel in the supervisor data area of CMem.

IOTBLSZ  I/O Table Size  CS20 [U-R, S-R/W]

The unsigned value in this register is the size in bytes of the I/O Table. If IOINDEX is greater than or equal to IOTBLSZ when an I/O instruction is executed, an IOERR fault occurs.

SWAP  SWAP Register  CS30 [U-R, S-R/W]

Any time the user/supervisor mode changes, either into or out of supervisor mode, ACU registers C30 and CS30 are swapped.

**Program Counter (PC)**

The 32-bit program counter indicates the currently executing instruction. The program counter is a byte counter into IMEM; however, all IMEM accesses must be 32-bit word aligned. The program counter’s value is only indirectly accessible to an ACU program. It is set by the jump and branch instructions. It is copied into register C31 by the jump subroutine instruction.

**CMem—ACU Memory**

CMem stores, in a byte-addressable fashion, all scalar program data for use by the ACU in a byte addressable fashion. The data format (big-endian or little-endian) matches the style of the FE. In contrast, IMem only stores program instructions.

CMem is divided into user and supervisor regions, as shown in Figure 2-2.

The CMem user region provides the only data storage location for scalar variables for use by user-mode programs on the ACU. The data layout within this region is completely user-defined. The CMem user region is accessible in both user and supervisor mode.
The CMem supervisor region is subdivided into areas serving a variety of purposes. The unifying characteristic is that the CMem supervisor region is accessible only in supervisor mode. If a user mode program attempts to access the CMem supervisor region, a CMem limit fault is generated. The CMem supervisor region contains the following areas:

**Direct-Mapped VME Space** The direct-mapped VME space area provides direct access into the corresponding addresses of the VMEbus address space. Reads and writes to the CMem VME region cause the ACU to read and write the corresponding addresses of the VME space. No address translation is provided; all CMem VME region requests are sent directly to the VMEbus at the corresponding address. For example, an ACU supervisor program can directly read and write VME peripherals such as an I/O Processor (IOP) using ACU load and store instructions. A write to an undefined region of the VME space produces a bus fault, which can be caught. By suitably synchronizing the machine, this action can be used to probe for the existence of a VME device.
Microcode Data
The microcode data area stores variables that are used exclusively by the MasPar microcode. If any values are changed in this area by a supervisor-mode program, undefined program operation occurs.

Supervisor Data
The supervisor data area contains data used by the ACU kernel routines and the parallel I/O instructions. Except for the data used to control the parallel I/O instructions (see I/O instruction description later on Page 3-31), the kernel program provides the only data interpretation.

Register Interrupt
The register interrupt area contains thirty-two 32-bit words that are directly mapped into VME bus addresses (see Page 5-10). The direct VME and CMem mapped locations provide a simple mechanism for data exchange between VME devices (such as the FE) and the ACU. The 32 locations are subdivided into four 8-register groups, where the 8-register set acts as one logical unit. The purpose of each unit is to provide a single communication channel between a VME device and the ACU kernel code. Each unit can be used to initiate interrupts between a VME device and the ACU, as well as to pass associated data. For example, one 8-register unit provides private communication between the the FE kernel and the ACU kernel. Two units are allocated for IOP-to-ACU kernel communication. The fourth unit provides communication between the active debugger and the ACU kernel.

Configuration Data
The configuration data area contains system information about the the machine. Examples include the sizes of the various memories, PMem, PReg, CMem, etc. Also included are serial numbers and board identification data. All such data is implementation dependent.

Trap Table
The trap table is a fixed data structure supporting 16 unique trap records. Each record contains four 32-bit entries. When any trap (such as a program fault, supervisor call, or interrupt) is encountered, the current PC and CPSW are saved in the first three words of the record. The final word in the 4-word record contains the starting IMem location of the service routine. As for all ACU instruction addresses, it must be 32-bit word aligned (two LSBs equal 0).

**PE Register/Memory Model**

Figure 2-3 shows the logical blocks inside the PEs:

PReg General purpose registers in each PE
SReg Thirty-two 32-bit registers in each PE that provide access to special PE status information, such as the PE Processor status word (PPSW)
Figure 2-3 PE Block Diagram

Accumulator A scratch register for use in aggregating and manipulating PE operands

PMem PE Memory containing user data, microcode data, and supervisor data

Since the PEs have a simple load/store style architecture, all PE computation occurs using only the general-purpose registers (PReg). The PE special registers contain special hardware status and control information like the PPSW. PMem is the bulk data storage area. The next sections describe each register in more detail.

PReg—PE General Purpose Registers

PReg is a high-performance register set for use by all PE instructions. The register set can be accessed for 1-, 8-, 16-, 32-, or 64-bit operands. No structure is imposed on PReg; however, it is typically used as 32-bit registers. PReg size is implementation dependent, but at least 160 bytes (1280 bits) are accessible in user mode.

SReg—PE Special Purpose Registers

Thirty-two 32-bit registers are contained in SReg. They are numbered S0-S31.

- S0 contains the PE Processor status word (PPSW).
- S1-S31 are implementation-dependent.
**LSB**

PE Accumulator LSB Bit 0 — [U-R/W]
This bit is the least significant bit of the PE accumulator. Depending on the last PE instruction executed, this bit might or might not be defined.

The four PE condition flags (C, V, Z, and N) summarize the last calculation result. The definition of each opcode specifies its exact interaction with these flags.

- **C-flag** PE Carry Flag Bit 1 — [U-R/W]
- **V-flag** PE Overflow Flag Bit 2 — [U-R/W]
- **Z-flag** PE Zero Flag Bit 3 — [U-R/W]
- **N-flag** PE Negative Flag Bit 4 — [U-R/W]
- **L-bit** PE Logical Accumulator Bit 5 — [U-R/W]
The L-bit stores intermediate results during single-bit calculations, including the calculation of boolean expressions that determine whether PEs participates in a computation.
- **E-bit** PE Enable Bit Bit 6 — [U-R/W]
The E-bit, if set, permits that PE to execute...
register-to-register instructions. The term "contingent execution" describes operations whose execution is contingent on the setting of the E-bit. There are significant exceptions to the rule that the E-bit controls PE compute instruction execution. These are noted with the description of the relevant instructions.

**M-bit**

PE Memory Access Bit  
Bit 7—[U-R/W]  
The M-bit, if set, permits that PE to execute instructions that transfer data between PE registers and memory. There are exceptions to the rule that the M-bit controls PE memory access. These are noted with the descriptions of the relevant instructions.

The three PE router status flags (T, R, and F) are used to coordinate the use of the global router. The definition of each opcode involving the router specifies its exact interaction with these flags. Note that a specific PE can send, receive, do both, or do neither during any given communication operation.

**T-flag**

PE Router Transmit Flag  
Bit 8—[U-R/W]  
This flag indicates that this PE wants to initiate router communication. When an open instruction is executed, some or all of the PEs with their T-flag set successfully gain global router connections. After a close instruction has been executed, PEs that have tried and failed to send a message still have the T-flag set. A program loop, sensitive to T-flag status, can ensure that all required communication is completed.

**R-flag**

PE Router Receive Flag  
Bit 9—[U-R/W]  
This flag indicates that this PE was the recipient of a message. Program logic can use the R-flag to determine which PEs need to store the received message (which necessarily is received in a register) into PMem, for example.

**F-flag**

PE Router Fetch Flag  
Bit 10—[U-R/W]  
This flag indicates that this PE successfully received data during a data fetch. As in the case of the R-flag, the F-flag can be used to determine which PEs need to store the fetched data.

The five PE floating point flags (FX, FI, FD, FU, and FV) summarize the results of previous FP instructions. (All FP instructions have names starting with "f".) The definition of each opcode specifies its exact interaction with these flags. These PPSW bits correspond to bits in the ACU FLTTCOD special register. Any instruction that sets one of these PPSW bits (regardless of the bit's previous value) also
sets the corresponding FLTCOD bit, if it is enabled by FLTMSK, and thereby causes a fault.

In IEEE FP mode, FP instructions only set the FP flags and never clear them. The FP flags in IEEE mode can only be cleared by explicitly moving a zero into them with a movl, movluc, or smov32 instruction.

In VAX FP mode, every FP instruction rewrites all the FP flags.

**FX-flag**

PE Floating-Point Inexact Flag

Bit 11—[U-R/W]

IEEE-754, section 7.5, defines the inexact exception, but for performance reasons this exception is not implemented. Therefore, this flag is currently reserved in IEEE FP mode.

VAX FP has no inexact exception, so this flag is reserved in VAX FP mode.

**FI-flag**

PE Floating-Point Invalid Flag

Bit 12—[U-R/W]

In IEEE FP mode, this flag is set when an invalid FP operand (as defined by the IEEE FP format) is received; it is also set when an invalid FP operation (as defined by section 7.1 of the IEEE-754 standard) occurs.

In VAX FP mode, this flag is set when an invalid FP operand (as defined by the VAX FP format) is received; it is cleared otherwise.

**FD-flag**

PE Floating-Point Divide By Zero Flag

Bit 13—[U-R/W]

In IEEE FP mode, this flag is set when an FP divide by 0 is attempted, unless the other operand is also 0. (IEEE-754, section 7.1-7.2, defines 0 divided by 0 as an invalid operation, not a divide by 0.)

In VAX FP mode, this flag is set when an FP divide by 0 is attempted; it is cleared otherwise.

**FU-flag**

PE Floating-Point Underflow Flag

Bit 14—[U-R/W]

In IEEE FP mode, this flag is set when an FP operation underflows (as defined by IEEE-754, section 7.4).

In VAX FP mode, this flag is set when an FP operation underflows; it is cleared otherwise.

**FV-flag**

PE Floating-Point Overflow Flag

Bit 15—[U-R/W]

In IEEE FP mode, this flag is set when an FP operation overflows (as defined by IEEE-754, section 7.3).
In VAX FP mode, this flag is set when an FP operation overflows; it is cleared otherwise.

Bits 16 to 20 contain PE error flags which are updated regardless of trap and fault masking. In user mode, these bits always read 0 and cannot be modified by the smo$\text{v}$ instruction (which ignores attempts to do so).

**XNET_ERR**

PE X-Network Parity Error  
Bit 16—[S-R/W]  
This bit is written by every xnet, xnetc, and xnetp instruction, with the following meanings.

1  
The PE detected a parity error.

0  
The PE detected no error. If any PE writes a 1, an XNETPARITY fault occurs (eventually, since an XNETPARITY fault is imprecise).

**RTR_ERR**

PE Router Parity Error  
Bit 17—[S-R/W]  
This bit is written by every instruction that uses the router (ropen, rosend, rsend, rf fetch, rf fetchc, and some executions of iord and iowrt). The bits have the following meanings.

1  
The PE detected a parity error.

0  
The PE detected no error. If any PE writes a 1, an RTRPARITY fault occurs (eventually, since an RTRPARITY fault is imprecise).

**ECC_HARD**

PMem Uncorrectable (Hard) ECC Error  
Bit 18—[S-R/W]  
This bit is set by the PMem access instructions (ld, ld$_{uc}$, ldsol, ld$_{nf}$, st, st$_{uc}$, and st$_{sol}$) in every PE that detects an uncorrectable memory error. This bit is never cleared by these instructions; it can only be cleared by an explicit smo$\text{v}$32 instruction in supervisor mode. If any PE sets ECC_HARD (even if it was already set), the UPMEMECC bit in PMSTAT is set, causing an M_Mach trap (eventually, since an M_Mach trap is imprecise).
ECC_SOFT

PMem Corrected (Soft) ECC Error Bit 19—[S-R/W]

This bit is set by the PMem access instructions (ld, ld_uc, ld_nf, ldsol, st, st_uc, stsol) in every PE that detects and corrects a memory error. This bit is never cleared by these instructions; it can only be cleared by an explicit smov32 instruction in supervisor mode. If any PE sets ECC_SOFT (even if it was already set), the SPMEMECC bit in PMSTAT is set. Depending on the value of PMEMECC (the PMem soft ECC count in supervisor CMem), TPMEMECC in PMSTAT can also be set, causing an M_Mach trap (eventually, since an M_Mach trap is imprecise).

PMEMLIMIT

PMem Address Limit Error Bit 20—[S-R/W]

This bit is written by every indirect PMem access instruction (ld_.Px.Pe, ld_uc.Px.Pe, ld_nf.Px.Pe, ldsol_.Px.Pe, st_.Pe.Px, st_uc.Pe.Px, and stsol_.Pe.Px).

The bits have the following meanings:

1

While in user mode, the PE attempted to access an address equal to or greater than PMEMLIMIT.

0

This error did not occur.

In supervisor mode, PMEMLIMIT is always written with 0 in all PEs by the above indirect PMem access instructions. If any PE writes a 1, a PMEMLIMIT fault occurs (eventually, since a PMEMLIMIT fault is imprecise).

Reserved

Reserved Bit 21-31—[0]

In user mode, these bits always read 0 and can not be modified by the smov instruction (which ignores attempts to do so).

CAUTION: In supervisor mode, programs must only write zeros into these bits.
PE Accumulator

The PE accumulator is a temporary holding register for the output of the PE ALU. When performing operations on 8-, 16-, 32-, or 64-bit operands, the accumulator adjusts to the correct size so that the accumulator MSB and LSB are the MSB and LSB of the operand.

NOTE: Many PE calculate instructions can use the accumulator as their source or destination operand, but no instruction may use the accumulator as both source and destination.

The encoding of bits inside the accumulator is implementation-dependent. It is generally an error to write to the accumulator using one size, then read from it using another: the results are undefined.

PMem—PE Memory

PMem stores all parallel program data for use by the PEs and is divided into user and supervisor regions. The PMem user region provides the only data storage location for parallel variables for use by user-mode programs on the DPU. The data layout within this region is completely user-defined; the architecture does not define any structures in the PMem user data region. The PMem user region is accessible in both user and supervisor mode.

The PMem supervisor region is accessible only in supervisor mode. If a user-mode program attempts to access the PMem supervisor region, a PMem limit fault is generated. The PMem supervisor region contains areas for microcode data and for supervisor data. If any values are changed in the microcode area by a supervisor-mode program, undefined program operation occurs. The PMem supervisor region contains no architecturally imposed data structures.
Chapter 3

The DPU Instruction Set

This chapter defines the instruction set design of the DPU. The first section describes the instruction formats. The second section provides a tabular description of the instruction set.

NOTE: It is not the intention of these tables to define either the syntax or mnemonics for the MasPar "machine code" or assembler input, but rather to enumerate the instructions that are understood by the ACU logic. The descriptions here do not necessarily summarize all the exceptional conditions, nor are timings provided.

The instruction categories are

- ACU Move/Load/Store Instructions
- ACU Calculate Instructions
- ACU Jump and Branch Instructions
- PE Move/Load/Store Instructions
  - PE Contingent and Uncontingent Flag Move Instructions
  - PE Condition Code Move Instructions
- PE Calculate Instructions
  - PE Integer Instructions
  - PE Boolean Instructions
  - PE Floating-Point Instructions
• PE Communication Instructions
  • PE Router Instructions
  • PE X-Network Instructions
• PE I/O Instructions
• ACU/PE Miscellaneous Instructions
  • ACU → PE Distribute Instruction
  • PE → ACU Global OR Instruction
  • PE Generate Address
  • Halt
  • No operation
  • PE Enable/Disable

DPU Instruction Description Notation

This manual uses the conventions described in the next two sections.

ACU References

CReg

CReg refers collectively to the ACU registers, which are individually named C0 through C31 and encoded 0 to 31 in instruction formats.

CSReg

CSReg refers collectively to the ACU Special registers, which are individually named CS0 through CS31 and encoded 0 to 31 in instruction formats.

CMem

CMem refers collectively to the ACU data store. CMem is byte addressed. CMem byte ordering is the same as that of the FE (on the VAX, the address is the lowest byte in the data item).

base

The term base means an ACU register that is used as a base register (in a base + offset calculation) in an instruction. Base registers can be used to reference CMem, PReg, and PMem. Specifically, in the instruction descriptions, CReg\textsubscript{base} means that base is a number between 0 and 31, and the contents of \textit{Cbase} are used in an address calculation.

NOTE: C0 always reads as 0 and is never changed by being written to.

CReg[...]

This means the contents of the identified ACU register. As noted above, a typical use is CReg\textsubscript{base}, where base identifies one of the 32 ACU registers, and CReg\textsubscript{base} means the contents of that register.
CSReg[...]
This means the contents of the identified ACU Special register.

Fb or Tb
This refers to the data queues from which data is received from the FE (TOBEQ) and to which data is sent to the FE (FRBEQ).

PE References

PReg
PReg-refers to the PE registers. The bit-addressed PReg can be referenced as if it were a byte-multiple, variable-length register. PE instruction variants are provided for dealing with 1-, 8-, 16-, 32-, and 64-bit operands. PReg is a bit-address of the low-order bit of the PReg memory to be used. For other than bit-operand instructions, the address so formed must be on a byte boundary in PReg; for example, the three low-order bits in the address must be zero or a hardware exception occurs.

The PE accumulator is specified as if it were PReg by the distinguished PReg bit offset 0x100 (for 9-bit PReg offset fields), or 0x4000 (for 15-bit PReg offset fields). If these special PReg offsets are used, reference to a base register in the instruction is ignored by the ACU.

Many bits of the PPSW can also be addressed using special PReg offsets as shown in Table 3-3.

PReg[...]
This means the contents of the identified PReg memory locations, in which the instruction is required to determine the length of the "register." For example, PReg[offset] means the contents of PE PReg beginning at bit offset, with the operand length given by the instruction.

SReg
SReg refers collectively to the PE special registers, which are individually named PS0 through PS31 and encoded 0 to 31 in instruction formats.

SReg[...]
This means the contents of the identified PE Reg.

PMem
PMem is a byte-addressed reference to the PMem associated with each PE. The length of the item referenced is specified by the instruction in which the reference occurs.

flg
flg refers to one of the one bit PE flags (see Table 3-3). Typically these flags are referenced by using a pseudo PReg address, which the ACU recognizes as a reference to the suitable flag. Typically flg (offset) is used to refer to the flag’s pseudo address.
### Addressing Modes

The instruction code tables that follow use the addressing mode abbreviations defined in Table 3-1.

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Name</th>
<th>Specifies</th>
<th>Base</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ac</td>
<td>address calculation</td>
<td>CReg[src/dst]+offset (15 bit)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Al</td>
<td>address calculation (long)</td>
<td>CReg[src/dst]+offset (32 bit)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Cm</td>
<td>CMem</td>
<td>CMem[ CReg_{base} + offset (15 bit) ]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Ce</td>
<td>CMem extended</td>
<td>CMem[ CReg_{base} + offset (32 bit) ]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Cr</td>
<td>CReg</td>
<td>CReg[src/dst]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Sr</td>
<td>CSReg</td>
<td>CSReg[src/dst]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Pr</td>
<td>PReg</td>
<td>PReg[ PReg_{base} + offset (9 bit) ]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Pe</td>
<td>PReg extended</td>
<td>PReg[ CReg_{base} + offset (15 bit) ]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Ps</td>
<td>PReg same base</td>
<td>PReg[ CReg_{same base as src} + offset (9 bit) ]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Pd</td>
<td>PReg direct</td>
<td>PReg[ offset (9 bit) ]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Pz</td>
<td>SReg</td>
<td>SReg[src/dst]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Px</td>
<td>PMem indirect</td>
<td>PMem[ PReg[ CReg_{base} + offset (15 bit) ]]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Pm</td>
<td>PMem</td>
<td>PMem[ CReg_{base} + offset (27 bit) ]</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Is</td>
<td>Immediate short</td>
<td>Offset (sign extended 9 bit)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Im</td>
<td>Immediate</td>
<td>Offset (sign extended 15 bit)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Ie</td>
<td>Immediate extended</td>
<td>Offset (32 bit)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Ib</td>
<td>Immediate big extended</td>
<td>Offset (64 bit)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Fb</td>
<td>(adaptor) FRBEQ</td>
<td></td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Tb</td>
<td>(adaptor) TOBEQ</td>
<td></td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>No</td>
<td>no mode</td>
<td>No addressing mode is used</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

**Table 3-1** Addressing Mode Abbreviations

**NOTE:**

All immediates are sign-extended to the operand size before performing a signed calculation.

For the \(\text{Px}\) mode, the PMem address in the specified PReg location is a 32-bit quantity.

The treatment of the offset value in base + offset calculations depends on the operand size.
• For one-byte operands, the offset and base are simply added.

• For multi-bit operands, the offset is
  1. shifted left by three bits
  2. added to the base

This reflects the fact that multi-bit operands must be byte-aligned, and extends the addressing range at the offset.

Table 3-2 shows valid immediate addressing modes, arranged by operand type and size:

<table>
<thead>
<tr>
<th>Operand Modes</th>
<th>Operand Type</th>
<th>Operand Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is</td>
<td>Integer</td>
<td>8, 16, 32, 64</td>
</tr>
<tr>
<td>Im</td>
<td>Integer</td>
<td>8, 16, 32, 64</td>
</tr>
<tr>
<td>Ie</td>
<td>Integer and Floating-point</td>
<td>16, 32, 64</td>
</tr>
<tr>
<td>Ib</td>
<td>Integer and Floating-point</td>
<td>64</td>
</tr>
</tbody>
</table>

**Table 3-2 Immediate Addressing Modes**

**Instruction Formats**

The DPU instruction formats are defined in Figure 3-1. In the description of individual instructions, the format is referenced in parentheses.

**NOTE:** The format of immediate fields is dependent on the front-end workstation, since the data types are based on those of the front-end workstation.
<table>
<thead>
<tr>
<th>Format 1 (1 word)</th>
<th>7</th>
<th>5</th>
<th>15</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>CReg</td>
<td>immed/offset</td>
<td>base</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 2 (1 word)</th>
<th>9</th>
<th>9</th>
<th>9</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>PReg/immed</td>
<td>offset</td>
<td>base</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 3 (1 word)</th>
<th>12</th>
<th>1</th>
<th>9</th>
<th>9</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>0</td>
<td>CReg</td>
<td>offset</td>
<td>base</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 4 (2 words)</th>
<th>12</th>
<th>15</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>dst-offset</td>
<td>base</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32-bit word</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 5 (3 words)</th>
<th>12</th>
<th>15</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>dst-offset</td>
<td>base</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32-bit word</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32-bit word</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 6 (1 word)</th>
<th>12</th>
<th>15</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>immed/offset</td>
<td>base</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 7 (2 words)</th>
<th>7</th>
<th>5</th>
<th>15</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>src-offset</td>
<td>sbase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 ..... 0</td>
<td>distance</td>
<td>dst-offset</td>
<td>dbase</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 8 (3 words)</th>
<th>12</th>
<th>15</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>target-offset</td>
<td>tbase</td>
<td></td>
</tr>
<tr>
<td>0 ..... 0</td>
<td>src-offset</td>
<td>sbase</td>
<td></td>
</tr>
<tr>
<td>0 ..... 0</td>
<td>dst-offset</td>
<td>dbase</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 9 (2 words)</th>
<th>12</th>
<th>1</th>
<th>5</th>
<th>9</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>0</td>
<td>CReg</td>
<td>0</td>
<td>base</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32-bit offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format 10 (2 words)</th>
<th>12</th>
<th>15</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>src-offset</td>
<td>sbase</td>
<td></td>
</tr>
<tr>
<td>offset</td>
<td>dbase</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Instruction Specifications

This manual uses a uniform instruction description format. Instructions are presented with periods separating instruction elements, in the following format:

<operation><operand length><src addressing mode><dest addressing mode>

where:

<operation> is what the instruction does.

- ACU instructions begin with c; for example, cadd for ACU add.
- PE instructions do not begin with a special identifying letter. The PE instruction for PE add is simply add.

<operand length> tells which specific instruction is meant, since most of the instructions have various operand length formats, and it is the instruction code that specifies the operand length.

<src addressing mode> is how the address of the source operand is constructed, according to Tables 3-1 and 3-2.

<dest addressing mode> is how the address of the destination operand is constructed, according to Tables 3-1 and 3-2.

Examples

Consider the following examples:

cmov32.Tb.Cr moves 32 bits (cmov32) from the TOBEQ (Tb) to the identified CReg(Cr). As the Addressing Mode table shows, Cr is an abbreviation for CReg[base].

ld8.Px.Pe moves one byte (ld8) indirectly from PMem (Px) to PReg (Pe). A base and offset is required for each operand (see Px and Pe in Figure 3-1).

Important General Rules

These general statements apply to almost all instructions and are mentioned in the descriptions of particular instructions only where there is an exception to the following rules:

E-bit contingent Most PE instructions are only executed if the E-bit of that PE is set. However, a uc addressing mode prefix specifies that the instruction is to be executed independently of the setting of the E-bit. For example, movluc.Im.Pr unconditionally moves one bit from an immediate field in the instruction (movluc) to the specified PReg (Im.Pr) on all PEs.
PE Register Overlaps Any instruction that uses PReg as both source and destination where the source is not equal to the destination but the locations overlap (e.g., mul32 0[C0], 8[C0]), results in undefined results.

This is the only restriction on general overlap. Therefore:

- Any or all PE operands of an instruction can be identical.
- The two source operands of a three-operand instruction (e.g., fmac) can overlap arbitrarily, provided neither of them overlaps or is identical to the destination.

Special PE Registers For some instructions, unique offset values allow the accumulator and individual bits of the PPSW to be specified. Except for the accumulator, these are all 1-bit flags that can be used for bit-operand instructions only. See Table 3-3 for details.

<table>
<thead>
<tr>
<th>9-bit offset</th>
<th>15-bit offset</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>0x4000</td>
<td>ACC</td>
<td>Accumulator</td>
</tr>
<tr>
<td>0x100</td>
<td>0x4000</td>
<td>LSB</td>
<td>LSB of the accumulator</td>
</tr>
<tr>
<td>0x101</td>
<td>0x4001</td>
<td>C-flag</td>
<td>ALU result carry</td>
</tr>
<tr>
<td>0x102</td>
<td>0x4002</td>
<td>V-flag</td>
<td>ALU result overflow</td>
</tr>
<tr>
<td>0x103</td>
<td>0x4003</td>
<td>Z-flag</td>
<td>ALU result zero</td>
</tr>
<tr>
<td>0x104</td>
<td>0x4004</td>
<td>N-flag</td>
<td>ALU result negative / MSB of accumulator</td>
</tr>
<tr>
<td>0x105</td>
<td>0x4005</td>
<td>L-bit</td>
<td>Logical accumulator</td>
</tr>
<tr>
<td>0x106</td>
<td>0x4006</td>
<td>E-bit</td>
<td>PE contingent execution for register operations</td>
</tr>
<tr>
<td>0x107</td>
<td>0x4007</td>
<td>M-bit</td>
<td>PE enable for load/store operations</td>
</tr>
<tr>
<td>0x108</td>
<td>0x4008</td>
<td>T-flag</td>
<td>Communications transmit flag</td>
</tr>
<tr>
<td>0x109</td>
<td>0x4009</td>
<td>F-flag</td>
<td>Communications fetch flag</td>
</tr>
<tr>
<td>0x10A</td>
<td>0x400A</td>
<td>R-flag</td>
<td>Communications receive flag</td>
</tr>
<tr>
<td>0x10B</td>
<td>0x400B</td>
<td>FX-flag</td>
<td>Floating-point result inexact</td>
</tr>
<tr>
<td>0x10C</td>
<td>0x400C</td>
<td>FI-flag</td>
<td>Floating-point invalid operand</td>
</tr>
<tr>
<td>0x10D</td>
<td>0x400D</td>
<td>FD-flag</td>
<td>Floating-point divide by zero</td>
</tr>
<tr>
<td>0x10E</td>
<td>0x400E</td>
<td>FU-flag</td>
<td>Floating-point result underflow</td>
</tr>
<tr>
<td>0x10F</td>
<td>0x400F</td>
<td>FV-flag</td>
<td>Floating-point result overflow</td>
</tr>
</tbody>
</table>

Table 3-3 PE Flags and Meanings

PReg Offsets The PReg offset field for all single-bit operand instructions specifies a bit address offset. The offset is shifted left three bits, then added to the base register to form a bit address into PReg.
In all other instructions (8, 16, 32 and 64 bits), the PReg offset field specifies a byte address offset. As just described, special offset values can be used with some instructions to access the accumulator and PPSW flags.

Notation

For clarity, classes of instructions are defined together. The common parts of the definitions for the class are given once. An underscore character is used to signify that the given definition applies, with the obvious variation, for multiple instructions.

- \( \text{Inst}_{s-} \) means that \( \text{Inst} \) requires a size.
- \( \text{Inst}_{m-} \) means that \( \text{Inst} \) requires a mode.
- \( \text{Inst}_{s,m-} \) means that \( \text{Inst} \) requires both size and a mode.

A variant PE arithmetic op exists for each operand length; each instruction addressing mode exists for each op variant. The operands and addressing modes are specified first. The instruction format that applies is shown in parentheses.

For example,

Sizes: \( 8, 16, 32, 64 \)


Special: Acc allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub(_{s,m-})</td>
<td>dst - src → dst</td>
</tr>
<tr>
<td>sub(_{r,s,m-})</td>
<td>src - dst → dst</td>
</tr>
<tr>
<td>mul(_{s,m-})</td>
<td>dst * src → dst</td>
</tr>
</tbody>
</table>

Table 3-4 PE Arithmetic Instructions

The underscore after the mnemonic signifies that a size or mode must be substituted as shown. Thus, the first line in the table actually defines 20 instructions (4 operand sizes and 5 addressing modes).

The numbers in parenthesis specify the instruction format numbers for each of the addressing modes. For example:

Mode: \( .Pr.Ps(2) \)

signifies that the \( \text{sub8}.Pr.Ps \) instruction uses the instruction format 2. If the Modes field is labeled "op specific", only the mode identified with each instruction is allowed. The Special field specifies whether the PE accumulator or PE Flags are valid for use with this instruction.
In the following sections, tables for each of the types of instructions are given. Each uses the previously specified notation.
# Condition Codes

Table 3-5 describes the condition codes used for both the PE and ACU.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
<th>Signed?</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq</td>
<td>equal</td>
<td>dst=src</td>
<td>n/a</td>
<td>Z</td>
</tr>
<tr>
<td>ge</td>
<td>greater than or equal</td>
<td>dst≥src</td>
<td>signed</td>
<td>N⊕V</td>
</tr>
<tr>
<td>gt</td>
<td>greater than</td>
<td>dst&gt;src</td>
<td>signed</td>
<td>(N⊕V)+Z</td>
</tr>
<tr>
<td>hi</td>
<td>higher</td>
<td>dst&gt;src</td>
<td>unsigned</td>
<td>(C)+Z</td>
</tr>
<tr>
<td>hs</td>
<td>higher or same</td>
<td>src≥dst</td>
<td>unsigned</td>
<td>C</td>
</tr>
<tr>
<td>le</td>
<td>less than or equal</td>
<td>dst≤src</td>
<td>signed</td>
<td>(N⊕V)+Z</td>
</tr>
<tr>
<td>lo</td>
<td>lower than</td>
<td>dst&lt;src</td>
<td>unsigned</td>
<td>C</td>
</tr>
<tr>
<td>ls</td>
<td>lower than or same</td>
<td>dst≤src</td>
<td>unsigned</td>
<td>C+Z</td>
</tr>
<tr>
<td>lt</td>
<td>less than</td>
<td>dst&lt;src</td>
<td>signed</td>
<td>N⊕V</td>
</tr>
<tr>
<td>ne</td>
<td>not equal</td>
<td>dst ≠ src</td>
<td>n/a</td>
<td>Z</td>
</tr>
<tr>
<td>pl</td>
<td>plus</td>
<td>dst≥0</td>
<td>signed</td>
<td>N</td>
</tr>
<tr>
<td>mi</td>
<td>minus</td>
<td>dst&lt;0</td>
<td>signed</td>
<td>N</td>
</tr>
<tr>
<td>vs</td>
<td>overflow</td>
<td>n/a</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>vc</td>
<td>no overflow</td>
<td>n/a</td>
<td></td>
<td>V</td>
</tr>
</tbody>
</table>

Table 3-5 Condition Code Names and Equations
ACU Move/Load/Store Instructions

- ACU Register Move Instructions
- ACU Register Load/Store Instructions

ACU Register Move Instructions

Sizes: 32


Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmov32.Cr.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>cmov32.Cr.Fb</td>
<td>src → dst</td>
</tr>
<tr>
<td>cmov32.Tb.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>cmov32.Ie.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>cmov32.Im.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>cmov32.Sr.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>cmov32.Cr.Sr</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

Table 3-6 ACU Register Move Instructions

cmov32.Cr.Fb attempts to write data from a CReg into the FRBEQ. The FE later reads the data from the FRBEQ. If the FRBEQ is not full, data is deposited, and ACU execution continues. If the FRBEQ is full, the ACU operation stalls until the FRBEQ is again not full. Pending PE PMem load and store instructions continue until completed if the ACU stalls.

cmov32.Tb.Cr attempts to read data from the FE processor via the TOBEQ and move it to a CReg. If the FE has previously written to the TOBEQ, the earliest data written is retrieved, and ACU operation continues. If the TOBEQ is empty, the ACU stalls until the FE writes data. Pending PE PMem load and store instructions continue until completed.

The addressing modes of the cmov32 instructions reflect their special functions, which is to move data between the general purpose and the special purpose ACU registers.
ACU Register Load/Store Instructions

Sizes: 32


Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>cld32.Cm.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>cld32.Ce.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>cst32.Cr.Cm</td>
<td>src → dst</td>
</tr>
<tr>
<td>cst32.Cr.Ce</td>
<td>src → dst</td>
</tr>
<tr>
<td>clea32.Ac.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>clea32.Al.Cr</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

Table 3-7 ACU Register Load/Store Instructions

CMem is a limited resource and is intended for storing constants and managing the ACU control stack, not for storing arbitrary scalar data. The clea instruction forms the effective address by combining the immediate constant with the specified source register. The effective address is stored in the destination register.
ACU Calculate Instructions

Sizes: 32

Modes: .Cr.Cr(3), .Im.Cr(6)
        .Ie.Cr(4) (except for shifts)

Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>cadd32m_</td>
<td>dst + src → dst</td>
</tr>
<tr>
<td>csub32m_</td>
<td>dst - src → dst</td>
</tr>
<tr>
<td>csbr32m_</td>
<td>src - dst → dst</td>
</tr>
<tr>
<td>cand32m_</td>
<td>dst &amp; src → dst</td>
</tr>
<tr>
<td>cor32m_</td>
<td>dst</td>
</tr>
<tr>
<td>cxor32m_</td>
<td>dst ^ src → dst</td>
</tr>
<tr>
<td>cshl132m_</td>
<td>dst &lt;&lt; src → dst left logical shift</td>
</tr>
<tr>
<td>cshla32m_</td>
<td>dst &lt;&lt; src → dst left arithmetic shift</td>
</tr>
<tr>
<td>cshr132m_</td>
<td>dst &gt;&gt; src → dst fill with 0</td>
</tr>
<tr>
<td>cshra32m_</td>
<td>dst &gt;&gt; src → dst fill with C flag</td>
</tr>
<tr>
<td>ceql32m_</td>
<td>src - dst sets condition flags</td>
</tr>
</tbody>
</table>

Table 3-8  ACU Calculate Instructions

ACU Jump and Branch Instructions

ACU jump and branch instructions determine the control flow of the ACU macrocode.

Jump- or branch-to subroutine instructions store the address (PC) of the instruction following the jump or branch.

- If a cjsr is executed at location $N$, when the jump is executed, $N + 8$ is stored in C31.
- If a bsr is executed at location $N$, when the jump is executed, $N + 4$ is stored in C31.

The return instruction is cjmp . Cr using C31.
ACU Unconditional Jump and Branch Instructions

Sizes: (none)  
Modes: (op specific) .Cr(3), .Ie(4), .Im(6)  
Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>cjmp.Ie</td>
<td>src → PC</td>
</tr>
<tr>
<td>cjmp.Cr</td>
<td>src → PC</td>
</tr>
<tr>
<td>cjsr.Ie</td>
<td>PC\text{_return} → CReg31; src → PC</td>
</tr>
<tr>
<td>cjsr.Cr</td>
<td>PC\text{_return} → CReg31; src → PC</td>
</tr>
<tr>
<td>bra.Im</td>
<td>src + PC → PC</td>
</tr>
<tr>
<td>bsr.Im</td>
<td>PC\text{_return} → CReg31; src + PC → PC</td>
</tr>
</tbody>
</table>

Table 3-9 ACU Unconditional Jump and Branch Instructions

The PC offsets specified in the source field of both relative branch instructions are word offsets from the location after the branch instruction. For example, a src with a value of 0 for a bra or bsr instruction causes a transfer to the instruction following the branch. The target location of the jump instructions are byte addresses. The PC value saved in C_{31} in jump-or branch-to subroutine instructions is always the address of the instruction after the jump or branch.

ACU Conditional Branch Instructions

Sizes: (none)  
Mode: .Im(6)  
Special: Acc not allowed, PE Flags not allowed  
Conditions: eq, gt, hi, hs, le, lo, ls, lt, ne, pl, mi, ge, vc, vs

In the instruction below, \text{c\_} is the location of the condition.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_{c_m_}</td>
<td>src + PC → PC</td>
</tr>
</tbody>
</table>

Table 3-10 ACU Conditional Jump and Branch Instruction

The PC offset specified in the source field of this instruction is a word offset from the location of the instruction after the branch. Thus, an instruction that loops by branching to itself has an offset field of $-1$. 
PE Move/Load/Store Instructions

- PE Register Move Instructions
- PE Special Register Move Instructions
- PE Flag Move Instructions
- PE Condition Code Move Instructions
- PE Register Load/Store Instructions

PE Register Move Instructions

Sizes: 8, 16, 32, 64
Special: Acc allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov32m_</td>
<td>src ⊂ dst</td>
</tr>
</tbody>
</table>

Table 3-11 PE Register Move Instructions

All PE register move instructions move an operand from the source to the destination. The differences are in the addressing modes:

- .Is.Pr(2) This one-word form takes a 9-bit immediate as the source and moves it to a base and 9-bit offset-specified destination.
- .Pd.Pr(2) This one-word form constructs the source address of the PReg location (without base) and the destination from base and offset.
- .Pr.Pd(2) In the dual of the above, the source has full base and offset, whereas the destination has offset only.
- .Pr.Ps(2) This one-word form requires the same base for both source and destination, permitting such operations as moves within the registers in use by a single virtual processor.
- .Ie.Pe(4) This two-word instruction takes either a 16- or 32-bit immediate and moves it into the PReg location specified by the base plus a 15-bit offset.
- .Ib.Pe(5) This three-word instruction takes a 64-bit immediate and moves it into the PReg location specified by the base plus a 15-bit offset.
- .Pe.Pe(7) This form includes base and 15-bit offset specifications for both source and destination, thus allowing access to any two PReg locations.
PE Special Register Move Instructions

Size: 32
Modes: .Pz.Pr(3), .Pr.Pz(3)
Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>smov32m_</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

Table 3-12 PE Special Register Move Instructions

As indicated by the addressing modes, these instructions move data between PReg and SReg in either direction.

PE Flag Move Instructions

Sizes: 1
Special: Acc not allowed, PE Flags allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov1m_</td>
<td>src → dst</td>
</tr>
<tr>
<td>movlucm_</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

Table 3-13 PE Flag Move Instructions

PReg→flg/flg→PReg instructions occur only for those PEs enabled for contingent execution (PEs with E-bit set). The uncontingent instruction variants (uc) are always executed by all PEs. See Table 3-3 for the definition of the flags.

PE Condition Code Move Instructions

The PE Condition Code instructions allow single-instruction operations that move one of the 12 condition codes (based on the flags C, V, N, and Z) into a PReg location. These instructions are useful for implementing programming language IF constructions before setting the E-bit to control contingent PE execution. The full instruction is specified by replacing the underscore in pb_1.Pe with the desired condition code. The syntax does not include a period before the condition code because the condition code is part of the opcode. All condition code instructions use format-6 instructions with all addresses treated as bit addresses. Flag destination specifiers are allowed, as with pbeq1.Pe.
In Table 3-14, c_ shows the location of the direction in the instructions.

All these instructions execute only if the E-bit is set.

**Conditions:**  
`eq, gt, hi, hs, le, lo, ls, lt, ne, pl, mi, ge, vs, vc`

**Sizes:**  
1

**Modes:**  
`.Pe(6)`

**Special:**  
Acc not allowed, PE Flags allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>pbc_1.Pe</td>
<td>condition-code → dst</td>
</tr>
</tbody>
</table>

**Table 3-14 PE Condition Code Move Instructions**

**PE Register Load/Store Instructions**

**Sizes:**  
8, 16, 32, 64

**Mode:**  
(op specific) `.Pe.Px(7), .Px.Pe(7), .Pm.Pe(10), .Pe.Pm(10)`

**Special:**  
Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>lds_.Pm.Pe</td>
<td>src → dst</td>
</tr>
<tr>
<td>lds_uc.Pm.Pe</td>
<td>src → dst</td>
</tr>
<tr>
<td>ldsol_.Pm.Pe</td>
<td>src → dst</td>
</tr>
<tr>
<td>lds_.Px.Pe</td>
<td>src → dst</td>
</tr>
<tr>
<td>lds_uc.Px.Pe</td>
<td>src → dst</td>
</tr>
<tr>
<td>ldsol_.Px.Pe</td>
<td>src → dst</td>
</tr>
<tr>
<td>sts_.Pe.Pm</td>
<td>src → dst</td>
</tr>
<tr>
<td>sts_uc.Pe.Pm</td>
<td>src → dst</td>
</tr>
<tr>
<td>stsol_.Pe.Pm</td>
<td>src → dst</td>
</tr>
<tr>
<td>sts_.Pe.Px</td>
<td>src → dst</td>
</tr>
<tr>
<td>stst_..Pm</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

**Table 3-15 PE Register Load/Store Instructions**
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>stsol_{s..Pe.Px}</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

Table 3-15 PE Register Load/Store Instructions (cont.)

ld_{s}/st_{s} instructions are unique in that they occur concurrently with the execution of other PE and ACU instructions. Transfers between PReg and PMem are only enabled for processors that have the M-bit set at the time the load or store instructions are executed. A queue manages the load/store requests; the requests are executed sequentially on a first-in, first-out basis. The active load or store is overlapped with other operations that do not access PMem. Operations are interlocked so that the results are the same as if the load or store had been executed completely when it first occurred. For example, an instruction that requires access to a PReg that is being loaded stalls until the load is complete.

ld_{uc}/st_{uc} instructions are like the ld_{s}/st_{s} instructions, but they transfer data between PReg and PMem in all PEs without regard to the status of the M-flag.

ld_{.Px.Pe} and st_{.Pe.Px} instructions transfer data to or from the PMem, with the 32-bit PMem address specified in a PE register, thereby enabling PEs to use data-sensitive PMem locations (an "indirect" reference or "addressing autonomy").

ld_{uc.Px.Pe} and st_{uc.Pe.Px} instructions are like the ld_{.Px.Pe} and st_{.Pe.Px} instructions, but they transfer data between PReg and PMem in all PEs without regard to the status of the M-flag.

The ld_{sol} and st_{sol} (load and store solitary) instructions are optimized for the important case when it is known that precisely one PE in a given cluster has its M-bit set. This is the case for the code designed to implement intrinsic functions such as reduction. If more than one PE per cluster has the M-bit set, the SOLERR bit in PMSTAT is set, and an M_Mach trap occurs.

PE load and store operations ignore the E-bit and use the M-bit instead.

**PE Calculate Instructions**

Each PE contains an accumulator (Acc), as well as a set of registers (PReg). Initially, all PE calculations involve the accumulator and one register. The result of the calculation is placed in the accumulator. This is shown below:

\[ \text{acc op src → acc} \]

However, at the instruction-set level, the PE behaves as if it were a two-address, register-to-register machine, as shown below:

\[ \text{dst op src → dst} \]
Each instruction leaves the result in the accumulator as a side-effect. Since there is a significant speed advantage in using the contents of the accumulator, the ACU recognizes the PReg offset $0\times100$ (for 9-bit PReg offsets) or $0\times4000$ (for 15-bit PReg offsets) as specifying the accumulator. It invokes bypass logic to avoid the normal PReg access.

In the following classes of instructions, the E-bit in the PE must be set for the instruction to have any effect on the PE state:

- PE Integer Instructions
- PE Boolean Instructions
- PE Floating-Point Instructions

These instruction classes are described in the following sections.

**PE Integer Instructions**

PE integer instructions generally leave the result in the PE accumulator as well as in the destination operand.

**PE Binary Integer Instructions**

Sizes: 8, 16, 32, 64

Modes: $\text{Is.Pr}(2), \text{Pr.Ps}(2), \text{Cr.Pr}(3)$

$\text{Ie.Pe}(4), \text{Ib.Pe}(5)$ (except for shifts)

Special: Acc allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>add$_{s_m}$</td>
<td>dst + src $\rightarrow$ dst</td>
</tr>
<tr>
<td>sub$_{s_m}$</td>
<td>dst - src $\rightarrow$ dst</td>
</tr>
<tr>
<td>sub$_{r_m}$</td>
<td>src - dst $\rightarrow$ dst</td>
</tr>
<tr>
<td>mul$_{s_m}$</td>
<td>dst * src $\rightarrow$ dst</td>
</tr>
<tr>
<td>mulu$_{s_m}$</td>
<td>dst * src $\rightarrow$ dst (unsigned)</td>
</tr>
<tr>
<td>div$_{s_m}$</td>
<td>dst / src $\rightarrow$ dst</td>
</tr>
<tr>
<td>divu$_{s_m}$</td>
<td>dst / src $\rightarrow$ dst (unsigned)</td>
</tr>
<tr>
<td>mod$_{s_m}$</td>
<td>dst % src $\rightarrow$ dst</td>
</tr>
<tr>
<td>modu$_{s_m}$</td>
<td>dst % src $\rightarrow$ dst (unsigned)</td>
</tr>
</tbody>
</table>

*Table 3-16 PE Binary Integer Instructions*
### PE Binary Integer Instructions (cont.)

#### PE Unary Integer Instructions

Sizes: 8, 16, 32, 64  
Modes: .Pe(6)  
Special: Acc allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>and&lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>dst &amp; src → dst</td>
</tr>
<tr>
<td>or&lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>dst</td>
</tr>
<tr>
<td>xor&lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>dst ^ src → dst</td>
</tr>
<tr>
<td>shll&lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>dst-&lt;&lt; src → dst (left logical shift)</td>
</tr>
<tr>
<td>shla&lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>dst &lt;&lt; src → dst (left arithmetic shift)</td>
</tr>
<tr>
<td>shr&lt;sub&gt;l&lt;/sub&gt; &lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>dst &gt;&gt;= src → dst (right logical shift)</td>
</tr>
<tr>
<td>shr&lt;sub&gt;a&lt;/sub&gt; &lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>dst &gt;&gt; src → dst (right arithmetic shift)</td>
</tr>
<tr>
<td>eql&lt;sub&gt;s_m&lt;/sub&gt;</td>
<td>src = dst (only sets condition flags)</td>
</tr>
</tbody>
</table>

#### PE Integer Conversion Instructions

These all have the form:

```
<type of conversion><fmt><fmt>
```

where `<fmt>` is `<size><type>`

The types are:

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>signed integer</td>
</tr>
<tr>
<td>u</td>
<td>unsigned integer</td>
</tr>
</tbody>
</table>

### PE Integer Conversion Types
In Table 3-19, the addressing mode has been left out to improve readability. The addressing mode is \_Pe.Pe in all cases. The instructions contain underscores for the sizes. The sizes for operands are 8, 16, 32, and 64. Note that "null op" combinations do not exist — for example, \texttt{pconv16s16s} is not an instruction.

**FP Sizes:** 32, 64  
**Integer Sizes:** 8, 16, 32, 64  
**Modes:** \_Pe.Pe(7)  
**Special:** Acc allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>pconv_{s_s_s}</td>
<td>src \to dst (with conversion)</td>
</tr>
<tr>
<td>pconv_{s_u_s}</td>
<td>src \to dst (with conversion)</td>
</tr>
<tr>
<td>pconv_{s_s_u}</td>
<td>src \to dst (with conversion)</td>
</tr>
<tr>
<td>pconv_{s_u_u}</td>
<td>src \to dst (with conversion)</td>
</tr>
</tbody>
</table>

**Table 3-19 PE Integer Conversion Instructions**

**PE Boolean Instructions**

All PE boolean instructions operate on single bits. Either the source or the destination can be one of the unique flag identifiers. All instructions are E-bit conditional (even if the E-bit is the destination). The PE Logic Accumulator and the instruction destination (L-bit) always get the result of these instructions.

**Sizes:** 1  
**Modes:** \_Is.Pr(2), \_Pr.Ps(2), \_Pe.Pe(7)  
**Special:** Acc not allowed, PE Flags allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>and_{m_}</td>
<td>dst &amp; src \to dst</td>
</tr>
<tr>
<td>or_{m_}</td>
<td>dst</td>
</tr>
<tr>
<td>xor_{m_}</td>
<td>dst \or src \to dst</td>
</tr>
</tbody>
</table>

**Table 3-20 PE Boolean Instructions**
The instruction shown in Table 3-25 is a unique PE boolean instruction that has unconditional operation (independent of the E-bit). This instruction is useful for the ELSE part of an IF/THEN/ELSE programming construct.

Sizes: 1
Special: Acc not allowed, PE Flags allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>else luc_m_</td>
<td>dst^src → dst</td>
</tr>
</tbody>
</table>

Table 3-21 PE Uncontingent Flow Control Instruction

PE Floating-Point Instructions

Floating-point binary and unary instructions are specified in the following sections. Floating-point instructions leave the PE accumulator with an undefined value.

PE Floating-Point Binary Instructions

Sizes: 32, 64
Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>fadd_s_m_</td>
<td>dst + src → dst</td>
</tr>
<tr>
<td>fsub_s_m_</td>
<td>dst - src → dst</td>
</tr>
<tr>
<td>fsubr_s_m_</td>
<td>src - dst → dst</td>
</tr>
<tr>
<td>fmul_s_m_</td>
<td>dst * src → dst</td>
</tr>
<tr>
<td>fdiv_s_m_</td>
<td>dst / src → dst</td>
</tr>
<tr>
<td>frem_s_m_</td>
<td>remainder(dst / src) → dst</td>
</tr>
<tr>
<td>feql_s_m_</td>
<td>src - dst (only sets the condition code)</td>
</tr>
</tbody>
</table>

Table 3-22 Floating-Point Binary Instructions
PE Floating-Point Unary Instructions

Sizes: 32, 64
Modes: (op specific) .Pr.Ps(2), .Pe(6)
Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsqrt.s_.Pr.Ps</td>
<td>sqrt(src) → dst</td>
</tr>
<tr>
<td>fneg.s_.Pr.Ps</td>
<td>- src → dst</td>
</tr>
<tr>
<td>ftst.s_.Pe</td>
<td>dst → dst (only sets condition flags)</td>
</tr>
</tbody>
</table>

Table 3-23  Floating-Point Unary Instructions

PE Floating-Point Operand Conversion Instructions

These all have the form

<type of conversion><fmt><fmt>

where <fmt> is <size><type>

The types are

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>floating</td>
</tr>
<tr>
<td>s</td>
<td>signed integer</td>
</tr>
</tbody>
</table>

Table 3-24  PE Floating-Point Conversion Types

In Table 3-25, the addressing mode has been left out to improve readability. The addressing mode is always .Pe.Pe. The instructions contain underscores for the sizes. The sizes for integers are 8, 16, 32, and 64; the sizes for floating-point are 32 and 64. Note that "null op" combinations do not exist — for example, fround16f16f is not an instruction.
FP Sizes: 32, 64
Integer Sizes: 8, 16, 32, 64
Modes: .Pe.Pe(7)
Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>fround_s_f_s_s</td>
<td>src → dst (with conversion)</td>
</tr>
<tr>
<td>ffloor_s_f_s_s</td>
<td>src → dst (with conversion)</td>
</tr>
<tr>
<td>ftrunc_s_f_s_s</td>
<td>src → dst (with conversion)</td>
</tr>
<tr>
<td>fceil_s_f_s_s</td>
<td>src → dst (with conversion)</td>
</tr>
<tr>
<td>fround_s_s_s_f</td>
<td>src → dst (with conversion)</td>
</tr>
<tr>
<td>fround_s_f_s_f</td>
<td>src → dst (with conversion)</td>
</tr>
</tbody>
</table>

Table 3-25 Floating-Point Conversion Instructions

PE Communication Instructions

Two PE-to-PE communication mechanisms are provided.

- The global router network, with which data messages can be transmitted in bit-serial format, one bit per clock cycle, from each PE to any other PE in the array.

- The X-Network, with which data can be exchanged between neighboring PEs, one bit per clock cycle.

Both X and router network communications are between physical PEs. Any virtual processing concept must be implemented by macrocode.

Router Network

The router network permits a PE to send a message to any other PE. This form of communication requires an explicit calculation of the target PE address in each PE initiating communication.

The actual communication of messages breaks into the following phases:

All PEs that want to communicate set their T-flags.

All PEs that have their T-flags set attempt to open a communication channel to a target PE.

For those that succeed, bit serial messages can be transmitted between the requester and the target (after the channel is open, messages can be sent in either
direction, under program control). Then the channel is released, and the T-flag is cleared.

Those PEs that fail (the interconnection switch can suffer blockage) wait until the current message transmission cycle is complete, then recontend for the channels.

The phases must be explicitly programmed using instructions described later in this section.

X-Network

The X-Network provides PE-to-PE communication using a two-dimensional mesh, in which each processor is directly connected to its eight nearest neighbors.

X-Network communications do not require PEs to have any explicit knowledge of which other PEs they are communicating with, since the communication is all geometrically relative.

The X-network is designed so that topologically the top of the array connects to the bottom of the array and the left side to the right, toroidally. Data transferred "north" from the top of the array arrives from the "south" at the bottom of the array; data transferred "east" from the right of the array arrives from the "west" at the left of the array.

Router Instructions

Four primitive instructions and two composite instructions are supplied. A brief description of each instruction follows:

- **Primitive Instructions**
  - `ropen` — attempt to open a communication channel to another PE (identified as described on p. 2-3)
  - `rsend` — send data on an open channel
  - `rfetch` — receive data on an open channel
  - `rclose` — close an open channel

- **Composite Instructions**
  - `rosend` — open and send (saves pipeline delay)
  - `rfetchc` — fetch and close (saves pipeline delay)

The status of the three communication PE flags determines the flow during communication:

- The T-flag indicates that the PE has unsatisfied communication needs. At the start of a communications macrocode loop, each PEs requiring global router communication must set its T-flag. The T-flag remains set until a PE has successfully transmitted data and executes the close instruction.
- The R-flag is set in a PE when a channel is opened to that processor. The set R-flag is used to condition the logic that causes that PE to perform receive-data cycles while the message is being transmitted. If the ACU programmer wants to transfer received data to PMem, then the R-flag would be unconditionally moved to the M-bit after a send to condition PE execution for the following store-to-memory instruction.

- The F-flag is set in all PEs that attempt to fetch data and succeed during a given communication cycle.

In Table 3-26, "sender" means the PE that opened the connection, and "receiver" means the PE at the other end of the connection. A PE can be a sender and a receiver at the same time.

Sizes: 1, 8, 16, 32, 64

Modes: (op specific) .No(3), .Pe(6), .Pe.Pe(7), .Pe.Pe.Pe(8)

Special: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ropen.Pr</td>
<td>opens a router connection to target PE</td>
</tr>
<tr>
<td>rosend_s_.Pe.Pe.Pe</td>
<td>opens a connection, then sends data: (sender) → dst(receiver)</td>
</tr>
<tr>
<td>rsend_s_.Pe.Pe</td>
<td>sends data: src(sender) → dst(receiver)</td>
</tr>
<tr>
<td>rfetch_s_.Pe.Pe</td>
<td>fetches data: src(receiver) → dst(sender)</td>
</tr>
<tr>
<td>rfetchc_s_.Pe.Pe</td>
<td>fetches data: src(receiver) → dst(sender), then closes connection</td>
</tr>
<tr>
<td>rclose.No</td>
<td>closes router connection</td>
</tr>
</tbody>
</table>

Table 3-26 Router Instructions
X-Network Instructions

Direction: N, NE, E, SE, S, SW, W, NW
Sizes: 1, 8, 16, 32, 64
Modes: .Pe.Pe.Cr(7)
Special: Acc not allowed, PE Flags not allowed

The PE interconnection wiring in the array is designed such that each PE can communicate with its eight nearest neighbors. In Table 3-27, d_ shows the location of the direction in the instructions.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xnet_d_s_m</td>
<td>src → dst</td>
<td></td>
</tr>
<tr>
<td>xnetp_d_s_m</td>
<td>src → dst</td>
<td></td>
</tr>
<tr>
<td>xnetc_d_s_m</td>
<td>src → dst</td>
<td></td>
</tr>
</tbody>
</table>

Table 3-27 X-Network Instructions

Figure 3-2 X-Network Geography

X-Network Instruction Descriptions

In all forms of X-Network Instructions:

- The instruction is SIMD—all PEs transfer data in the same, single direction.
- Only PEs with the T-flag set transmit data.
- Individual PEs can transmit data, receive data, do neither, or do both on a given instruction, depending on the instruction and the configuration of T-flags in the array.
• Data of the length specified in the instruction is transmitted (1, 8, 16, 32 or 64 bits).

• Data is transmitted in the specified direction, for the distance specified. For example, transmitting data to the east a distance of one PE sends data from each transmitting PE to the PE neighbor on the east. Transmissions that cross the "edge" of the array, loop around "toroidally." For example, transmissions sent to the east of the easternmost PE, arrive at the westernmost PE of the same row, as if they had come from the west.

• PEs that are between a transmitter and receiver (the receiver is the PE at the specified direction and distance from the transmitter) act as "passthrough" stages and pass the data on its way.

• PEs that receive data have their R-flag set. The specifics of receiving data are subtle, and the programmer should pay special attention to the characteristics of each instruction. PEs at the specified distance from a transmitting PE (one with the T-flag set) receive data, independently of the setting of the T-flag, but subject to the subtleties to be described.

• PEs that do not receive data have their R-flag cleared.

• An instruction with a distance of zero is comparable to a mov instruction, moving data from one PE register to another, with the R-flag set in all transmitting/receiving PEs (those with the T-flag set).

The xnetp and xnetc instructions modify the PE accumulator. For PEs with an R-flag set, the accumulator contains the received data. For other PEs, the accumulator contents is unspecified.

The characteristics of the instructions are as follows:

xnet  This instruction enables any collection of PEs to transmit (set their T-flags, before executing xnet) and each targeted receiving PE (each PE the designated distance from a transmitting PE) to receive the transmitted data. This instruction is the slowest X-Network instruction.

xnetp  This the "pipelined" version of the xnet instruction. It is much faster than xnet. It differs from xnet; with xnetp, when one of the passthrough stages between the transmitter and the receiver is also a transmitter, the transmitting stage blocks the message from being passed rather than also acting as a passthrough stage. The message that is blocked is not received by the target receiver, and the R-flag of that target receiver is not set.

xnetc  The xnetc instruction acts essentially like the xnetp instruction, suffers the same "blockage" problem, but, in addition, leaves a copy of the transmitted message in the intermediate passthrough PEs as well.

• In the simplest case, in which there are no passthrough stages also trying to transmit, that description is complete—the passthrough PEs also retain a copy of the transmitted data.
• In the case in which there is a passthrough PE that is also transmitting data, the effect upstream of the blockage (toward the transmitter) and downstream from the blockage (toward the receiver) is as follows:
  • Upstream (between the transmitter and the blocker) the passthrough PEs receive a copy of the transmitted data, oblivious to the fact that the target receiver does not receive the data.
  • Downstream from the blocker, the passthrough PEs receive and copy the blocker’s data, not the upstream transmitter’s data. The transmitting PE does not receive a copy of its own data unless the distance of the specified instruction is zero.

X-Network Examples

Consider the instruction \texttt{xnetE8.Pe.Pe.Cr} — move data on the X-Network, toward the east.

• The first PReg address specifies the operand source.
• The second PReg address specifies the destination.
• The CReg specifier identifies the ACU CReg that holds the “distance”—the number of PE stages through which data is transferred, along rows or columns or along diagonals.

Assuming that CReg contains the value 10, the following happens:

• For all PEs with their T-flag set, this instruction moves the specified single byte operand to the PE located 10 to the east and sets the R-flags of the corresponding PEs receiving the data.
• For PEs that are not receiving data, the destination location is not changed, and the R-flag is cleared.

This instruction requires roughly 10 * 8 clocks (distance * operand size) plus an initiation overhead.

When the pipelined version of X-Network communication is used, the T-flags must be set as follows:

• PEs sending data have their T-flags set.
• PEs acting as pipeline connection stages have their T-flags cleared.

Consider the use of the pipelined version of the same instruction to transfer data between every 10th PE, with the rest operating as pipeline stages only.

• CReg is initialized with the value 10.
• Code sets the T-flags in PEs in each 10th column (0, 10, 20, ...) and clears the T-flags in all others.
• The instruction executed is `xnetpE8.Pe.Pe.Cr.`, which moves an 8-bit operand in pipelined mode to the east.

The instruction moves the source operand in PEs to the destination in the T-flag enabled PEs 10 columns to the east. The T-flag disabled PEs act as pipeline stages.

Whereas execution time of the nonpipelined version is proportional to:

\[(\text{distance} \times \text{length}) + k\]

clock cycles, the pipelined version is proportional to:

\[\text{distance} + \text{length} + k\]
clocks, which can be a significant difference.

### PE I/O Instructions

**Sizes:** 8, 16, 32, 64  
**Modes:** `Pe.Pe.Cr(7)`  
**Special:** Acc allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iord_s_i</code></td>
<td>src → dst</td>
</tr>
<tr>
<td><code>iowrts</code></td>
<td>src → dst</td>
</tr>
</tbody>
</table>

#### Table 3-28  PE I/O Instructions

The I/O instructions move data between PReg and a memory-mapped I/O device. The device and its mapping are specified by IOINDEX (CS18), which is an index into the I/O Table. This table defines the available I/O devices and mappings. Each enabled PE contains a 32-bit offset into the memory-mapped region. The PE transfers data between its PReg and the location in the memory-mapped region identified by its own offset. This allows all PEs to independently access different addresses in the mapped region.

The CReg operand specifies the number of bytes to transfer per PE. The instruction size (8, 16, 32, or 64) specifies the natural size of the values being moved, which can be different than the CReg byte-count operand if multiple values are being moved for each PE. The instruction size is significant only if the I/O device performs automatic byte swapping.
Miscellaneous Instructions

These are assorted control and mode instructions.

**ACU ↔ PE Move Instructions**

Sizes: 1


Special: Acc not allowed, PE Flags allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>gor1.Pr.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>gor1.Pe.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>dist1.Cr.Pr</td>
<td>src → dst</td>
</tr>
<tr>
<td>dist1.Cr.Pe</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

Table 3-29  Single-bit ACU ↔ PE Move Instructions

Sizes: 8, 16, 32

Modes: (op specific) .Pr.Cr(3), .Cr.Pr(3)

Special: Acc allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>gor_s_.Pr.Cr</td>
<td>src → dst</td>
</tr>
<tr>
<td>dist_s_.Cr.Pr</td>
<td>src → dst</td>
</tr>
</tbody>
</table>

Table 3-30  Multi-bit ACU ↔ PE Move Instructions

The gor (global OR) instruction logically ORs the requested value for all E-bit enabled processors. Data from a single PE can be moved by disabling all other processors. Alternatively, summary data from many PEs can be retrieved in OR-merged format. If no PE is enabled, the result is zero.

The dist (distribute) instruction transfers data from the identified ACU register to all PEs that have their E-bit set. If no PE is enabled, the result is a NOP.
Other Miscellaneous Instructions

Sizes: (op specific)
Modes: (op specific) .Cr(3), .No(3), .Ie(4), .Im(6), .Pe(6)

Special: graddr only: Acc allowed, PE Flags not allowed
All others: Acc not allowed, PE Flags not allowed

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operand Size</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>nop.No</td>
<td></td>
<td>no operation</td>
</tr>
<tr>
<td>graddr.Pe</td>
<td>32</td>
<td>physical PE address (row,col) → dst</td>
</tr>
<tr>
<td>enable.Im</td>
<td></td>
<td>enable PE specified by the 15-bit immediate operand</td>
</tr>
<tr>
<td>enable.Ie</td>
<td></td>
<td>enable PE specified by the 32-bit immediate operand</td>
</tr>
<tr>
<td>enable.Cr</td>
<td></td>
<td>enable PE specified in the CReg</td>
</tr>
<tr>
<td>disable.Im</td>
<td></td>
<td>disable PE specified by the 15-bit immediate operand</td>
</tr>
<tr>
<td>disable.Ie</td>
<td></td>
<td>disable PE specified by the 32-bit immediate operand</td>
</tr>
<tr>
<td>disable.Cr</td>
<td></td>
<td>disable PE specified in the CReg</td>
</tr>
</tbody>
</table>

Table 3-31  Miscellaneous Instructions

graddr

The graddr instruction moves the array address of the enabled PEs into the specified PReg, in the canonical address format defined previously on page 2-3.

enable

The processor specified by the source operand in the format defined on page 2-3 has its E-bit set. No other state in this PE is altered. No other PE is altered.

disable

The processor specified by the source operand in the format defined on page 2-3 has its E-bit cleared. No other state in this PE is altered. No other PE is altered.
Chapter 4

User/Supervisor Mode Operation

The user and supervisor modes allow an execution model supporting multiuser operation. A protection mechanism separates user program data from privileged kernel program data, even though the data resides in the same address space. This mechanism allows the kernel to manage multiple user programs and to prevent each individual user program from altering any program state other than its own.

The basic difference between user and supervisor operation is that supervisor-mode operation allows access to more machine resources, both in the ACU and in the PE array. The following are the key additional capabilities in supervisor mode:

- Access to configuration data that affects machine state
- Access to VME space
- Access to parallel I/O configuration and control data
- Execution of privileged instructions that affect the entire machine state
ACU Data Access

Figure 4-1 shows the CMem user and supervisor regions.

```
  +------------------+
  | 0xFFFF FFFF     |
  | Direct mapped   |
  | VME space       |
  +------------------+
  | Microcode data  |
  +------------------+
  | Supervisor data |
  +------------------+
  | Register interrupts |
  +------------------+
  | Configuration data |
  +------------------+
  | Trap table       |
  +------------------+
  | Microcode data  |
  +------------------+
  | User data        |
  +------------------+
  | 0x0000 0000      |
```

Figure 4-1 CMem Regions

If a user-mode program attempts to access any CMem location outside the user region, a CMem limit fault is generated. This is enforced by checking the addresses of all load/store operations against a limit value (CMEMLIMIT) stored in an ACU special register. This limit can only be modified in supervisor mode.

In supervisor mode, all CMem accesses are legal.
PE Data Access

In the PE array, the distinction between user and supervisor mode is similar to the ACU. PMem is separated into user and supervisor regions. If a user-mode program attempts to access any PMem location outside the user region, a PMem limit fault is generated. This is enforced by checking the addresses of all load/store operations against a limit value (PMEMLIMIT) stored in an ACU special register. This limit can only be modified in supervisor mode.

In supervisor mode, all PMem accesses are legal.

Limit checking is also done on PReg accesses, using a limit value (PREGLIMIT) stored in an ACU special register. If a user-mode program attempts to access any PReg location above this limit value, a PReg limit fault is generated. This limit can only be modified in supervisor mode.

In supervisor mode, all PReg access are legal.

ACU Instruction Access

IMem cannot be modified by any ACU program, due to the ACU’s Harvard-style architecture in which data and program memory areas are separate. Therefore, user- or supervisor-mode protection is not relevant to IMem.

Privileged Instructions

All instructions can execute in supervisor mode, and only some can be executed in user mode. If a supervisor-mode-only instruction is executed in user mode, an ILLOP fault is generated (see page 5-7). The supervisor-only instructions are

- **halt**: Stops the ACU. See Chapter 6 for details.
- **crft**: Return from Trap—Returns from processing the current trap (e.g., an interrupt or system call)
- **cirq**: Interrupt Request—generates a VME bus interrupt

A number of instructions have different behavior, depending on user or supervisor modes. Except for ld_nf, all load and store instructions (cld, cst, ld, ld_uc, st, st uc) in user mode are protected, only allowing data read and write access in the ACU and PE user regions. All other accesses receive a memory limit violation. However, in supervisor mode all accesses are legal, and no range checking is performed. The only other instructions where the behavior is different are the smov and csmov instructions where certain special registers are only accessible when in supervisor mode.
PE I/O Access

The DPU supports parallel PE I/O using the \texttt{IORD} and \texttt{IOWRT} Instructions. These user-mode instructions allow direct access to memory-mapped I/O devices (such as the I/O RAM or VME devices). User data on I/O devices is protected from unauthorized access by an I/O table controlled by the ACU kernel. The PE I/O instructions can only access the portions of memory-mapped I/O devices that are allowed by the I/O table.

The I/O table is stored in the supervisor region of CMem. Its location (\texttt{IOTBLBASE}) and size (\texttt{IOTBLSZ}) are specified by two ACU special registers that can only be modified in supervisor mode. The form and content of the I/O table are implementation dependent.

User-mode PE I/O requires close cooperation between the user-mode program and the supervisor-mode ACU kernel. This is because the PE I/O instructions cannot access any data not specified by the I/O table, which can only be modified in supervisor mode. This cooperation is orchestrated with the \texttt{cshall} instruction; the user-mode program uses \texttt{cshall} instructions to ask the kernel to modify the I/O table.

Changing User/Supervisor Mode

Supervisor-mode status is indicated by a bit in the CPSW. The supervisor-mode status bit can be modified as a result of the following events:

- executing one of these instructions:

  \begin{itemize}
    \item \texttt{csmov} \quad Moving a value into the CPSW can clear the supervisor mode bit
    \item \texttt{crft} \quad The return from trap instruction restores the previous CPSW; this might change the mode.
    \item \texttt{cshall} \quad The system call instruction always causes a trap into supervisor mode to provide a system service.
    \item \texttt{cbreak} \quad The break point instruction always causes a trap into supervisor mode to provide a debugger service.
  \end{itemize}

  Single-step operation traps to supervisor mode to provide this debugger service.

  - system fault (divide by zero or a memory parity error) traps to supervisor mode to handle the problem
  - machine reset, which initializes the machine and places it in supervisor mode
  - hardware interrupt traps to supervisor mode to service the interrupting device.

The \texttt{CPSW} can be updated by the \texttt{csmov} instruction, allowing the supervisor-mode bit to be changed. However, the supervisor-mode bit can only be modified by the \texttt{csmov}
instruction if the program is already in supervisor mode. This means the `csmov`
instruction can only be used for the transition from supervisor to user mode and never for
the reverse transition. All other transitions between user and supervisor modes are
accomplished with traps, as listed at the beginning of this section and described in detail
in the next chapter.
Chapter 5

Trap Processing

Traps are service requests that break normal program flow so that the ACU can service some immediate need. All traps cause the system to follow the same general sequence:

1. It suspends normal operation and enters supervisor mode
2. It identifies the trap and jumps to a unique handler
3. After the handler completes its task, the system resumes normal operation.

Traps fall into one of five basic categories:

System Calls: These instructions provide user programs with a mechanism for requesting kernel services, such as machine configuration information and I/O support.

Hardware Interrupts: These allow devices like the FE or an I/O processor to signal state changes to the ACU.

Faults: This category includes hardware problems such as parity errors and software problems such as divide by zero and address limit violations.

Breakpoints: The debugger uses this instruction to reenter the debugger and stop the program being debugged when it reaches the breakpoint instruction.

Single Step Events: The trap architecture is used to implement single-step debugging operation.

All traps have two other characteristics: they are either maskable or nonmaskable and precise or imprecise.
• Maskable traps can be masked by the system code; when the system receives a request for a trap that is masked, the request is denied, and the trap is not taken.

• Precise traps are those that can be attributed to a single instruction.

• Imprecise traps are those that cannot be attributed to a single instruction.

All traps are taken between instructions, as follows:

• A precise trap is taken immediately after the instruction that caused it.

• An imprecise trap can be taken after any instruction; therefore, there is no way to determine which instruction caused the trap.

Unless the trap handler arranges otherwise, system operation resumes after trap processing with the instruction that would have been executed next if the trap had not occurred. In general, when an imprecise trap occurs during user-mode program execution, it means that there is an unrecoverable error, and the user program terminates.

These are the key features of traps:

Breakpoints                      precise and non-maskable
Hardware interrupts            precise and maskable
M-Machine traps                imprecise and maskable
Single step events             precise and nonmaskable
System calls                   precise and non-maskable
System faults                  can be precise or imprecise and are nonmaskable. Although system faults are not maskable as a group, they can be individually masked as described in the section titled "Faults," page 5-7.

**Trap Processing Data Structures**

Two different data structures, the trap control record and the trap table, control trap processing.

• The trap table contains a unique entry for every type of trap.

• The trap control record shows what traps are pending and what traps are masked.

The interrupt disable (ID) bit of the CPSW also influences trap handling: no maskable traps are taken while it is set.
Trap Control Record

Two CSReg registers, TRAPREQ(CS14) and TRAPMSK (CS15), comprise the trap control record. TRAPREQ contains a bit corresponding to each (maskable) trap type, and TRAPMSK contains a bit corresponding to each bit in TRAPREQ.

TRAPREQ drives the maskable traps. When a condition occurs corresponding to any valid bit in TRAPREQ, that bit is set. The corresponding trap occurs immediately, unless that particular trap is masked in TRAPMSK, or all maskable traps are disabled by the ID bit in the CPSW.

Each TRAPREQ bit is automatically cleared when its corresponding trap is taken. Therefore, TRAPREQ can be read to see if any conditions corresponding to maskable traps have occurred without a trap having been taken. TRAPREQ cannot be explicitly modified by the csmov instruction, even in supervisor mode.

A TRAPREQ bit might be set but the corresponding trap might not be taken due to the values of ID and TRAPMSK. In that case, the trap is taken whenever these values are modified to allow the trap.

Note that the nonmaskable traps are handled completely independently of the trap control record.

See Chapter 2 of this manual for details on TRAPREQ, TRAPMSK, and the CPSW.
Trap Table

The trap table occupies a block of supervisor CMem. It contains sixteen records, numbered 0 through 15. Each record corresponds to a particular type of trap; the record number is also the index into the trap table. Table 5-1 shows the trap table assignments.

<table>
<thead>
<tr>
<th>Trap</th>
<th>Maskable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N</td>
<td>System reset - either a hardware reset or cold boot</td>
</tr>
<tr>
<td>1</td>
<td>N</td>
<td>Faults - either software or hardware faults</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>Unused</td>
</tr>
<tr>
<td>3</td>
<td>N</td>
<td>ccall - a system call instruction</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>cbreak - a system breakpoint instruction</td>
</tr>
<tr>
<td>5</td>
<td>N</td>
<td>Single Step - used by the debugger</td>
</tr>
<tr>
<td>6</td>
<td>N</td>
<td>Unused</td>
</tr>
<tr>
<td>7</td>
<td>N</td>
<td>Unused</td>
</tr>
<tr>
<td>8</td>
<td>Y</td>
<td>CTRL0A - a VME HW interrupt</td>
</tr>
<tr>
<td>9</td>
<td>Y</td>
<td>CTRL0B - a VME HW interrupt</td>
</tr>
<tr>
<td>10</td>
<td>Y</td>
<td>CTRL0C - a VME HW interrupt</td>
</tr>
<tr>
<td>11</td>
<td>Y</td>
<td>CTRL0D - a VME HW interrupt</td>
</tr>
<tr>
<td>12</td>
<td>Y</td>
<td>M_Mach - PE memory access</td>
</tr>
<tr>
<td>13</td>
<td>Y</td>
<td>Unused</td>
</tr>
<tr>
<td>14</td>
<td>Y</td>
<td>Unused</td>
</tr>
<tr>
<td>15</td>
<td>Y</td>
<td>Unused</td>
</tr>
</tbody>
</table>

Table 5-1  Trap Table Assignments

Each trap record contains the following information:

PC_ The first word stores the PC location where processing continues after the trap processing completes. Because traps are always taken between instructions, this is the address of the instruction that would have been executed immediately after the last instruction executed before the trap. When the trap instruction is executed, the value in PC_ is moved into the program counter transferring execution to that location. The PC_ contents are changed only by a CMem store instruction or by a trap at that particular entry. The trap instruction does not affect the stored value.

PSW_ The second word stores the CPSW at the time just before the start of trap processing. The stored CPSW value is used to restore the machine state to the pretrap condition by the trap instruction. The PSW_ contents are changed only by a CMem store instruction or trap to that particular entry point. The trap instruction does not effect the stored value.
Reserved: The third word is reserved and must not be used.

JMP: The fourth word provides the starting instruction location for processing the trap. After a trap has been detected by the ACU, program execution jumps to the location specified by the trap's JMP value after the PC and PSW values are updated. All used JMP locations must be set properly before any traps are processed.

Because only a single trap record is provided for each entry point, the architecture does not directly support reentrant trap routines. For example, the ccall trap handler cannot use ccall.

**General Trap Programming**

This section describes the basic details and processing steps used for processing traps. Subsequent sections describe each type of trap in detail, including any exceptions to the generic trap processing rules described in this section. The following paragraphs describe the generic trap processing sequence.

1. One or more trap requests occur. Each maskable trap request sets its corresponding TRAPREQ bit. The description of each trap type indicates the possible causes of that trap. In addition, a trap request is also considered to occur when a maskable trap request that was not allowed by the values of ID and TRAPMSK becomes allowed due to a change in the ID or TRAPMSK value.

2. The current instruction completes.

3. Traps are selected for handling. If any nonmaskable trap requests are present, they are all selected, and no other trap requests are considered. Otherwise, if ID allows maskable traps, then the lowest numbered maskable trap request allowed by TRAPMSK is selected and no other trap requests are considered. If no trap request is selected, no trap is taken; processing continues as if the trap request had not occurred, except that any set TRAPREQ bits remain set.

4. If the CPSW User/Supervisor-mode (US) bit indicates user mode, C30 and CS30 are swapped.

5. For each selected trap request, from highest to lowest numbered trap type, the following operations are performed:
   a. The current PC (the location of the instruction that would have been executed next if the trap request had not occurred) and CPSW are stored into the trap type's trap table entry.

---

1There are two instructions that can be stopped before completion by a trap: mov.Tb.Cr and mov.Cr.Fb (the queue access instructions). These are the only two instructions in the architecture that can take an unbounded time. Therefore, a nonmaskable trap (or a maskable trap allowed by the ID and TRAPMSK values) that is received while the machine is stalled on one of these instructions 'undoes' the instruction so that it appears that the instruction had never started. The trap then enters the trap handler. The MOV instruction is restarted after trap processing completes.
b. The CPSW is updated to enable supervisor mode, disable
single-step traps, and disable all maskable traps (i.e., interrupts).

c. The value of the JMP_ location in the trap type’s Trap Table entry
is copied into the PC. If this was a maskable trap, its trap request bit
is cleared in TRAPREQ. If there is more than one selected trap
request, they are said to be "threaded" by this procedure. Only
nonmaskable traps are threaded; maskable traps are only selected
one at a time.

6. The effect of the procedure just described for handling multiple traps is to
handle traps in priority order, from lowest to highest trap type number, with
all nonmaskable traps having higher priority than any maskable traps.

7. Execution begins in the trap handler now pointed to by the PC, allowing the
trap handler to process the request.

8. After the trap handler processes the request, the handler exits using the
crft instruction. The crft instruction restores the PC and CPSW from
the Trap Table. (The restored PC can point to a trap handler, if threaded
trap requests remain to be handled, or to the user program.)

9. If crft changed the user/supervisor mode, C30 and CS30 are swapped.

10. As a result of executing the crft instruction, all pending PE load and store
operations are completed.

11. Execution begins at the instruction pointed to by the PC. Unless this is a
threaded trap handler, normal program execution continues.

Based on this processing sequence, the following are key points:

- The Trap Table JMP_ locations must be initialized before any trap occurs,
  otherwise, the system jumps to some random location. This typically is
  accomplished during machine initialization.

- All trap handlers begin execution in supervisor mode, allowing access to all
data: user, supervisor, and even microcode. There are no memory limit
faults in supervisor mode.

- Both single-step and maskable traps are disabled at the start of a trap
  handler.

- Only the PC and CPSW are saved on entry into the interrupt handler. All
  other states that a program needs must be saved and restored explicitly by
  that program.

- The trap handler starts execution with all maskable traps disabled, which
  means that the processor can start processing a critical code section. If the
  entire handler does not need to be in a critical code section, maskable traps
  should be re-enabled in the CPSW.

- Handlers exit using the crft instruction, which restores the pre-trap PC and
  CPSW.
The following sections describe each of the trap types, as specified in the trap table.

Reset Operation

A reset trap can be initiated by either a machine cold start or by a VME device that requests a machine reset operation using the execution control and status register (ECSR). The reset trap sequence is nearly identical to all other trap processing. The following occurs:

1. The machine unconditionally and immediately stops processing the current instruction. It does not wait for the instruction to complete.

2. All pending PE load and store operations are immediately terminated.

3. The current PC is stored into the trap table. The PC can be pointing to the middle of an instruction. Any other trap requests are ignored; no trap threading is done.

4. The current CPSW is stored in the trap table.

5. The CPSW is updated, enabling supervisor mode, disabling single-step traps, and disabling all maskable traps.

6. The JMP value is not used, and the PC is always set to 0x81000000.

7. ECSR<RUNENBL> is cleared and the DPU halts.

8. If the FE sets ECSR<RUNENBL> without changing the PC or other DPU state, the DPU begins executing the reset trap handler.

Normally, a machine reset operation is only used during a cold start sequence, such as when the machine is powered on. Otherwise, the reset sequence is only necessary to recover from either a hardware or software system failure.

The machine reset trap is imprecise and not maskable. Instruction execution cannot be reliably continued.

Faults

Faults include both hardware and software problems. Hardware problems include parity errors and bus errors. Software problems include a wide variety of program errors such as:

- Numerical errors (floating point exceptions, divide by zero, etc.)
- Limit errors (memory address too large)
- Alignment errors (illegal instruction address)
• Illegal instruction

Precise faults (such as a divide by zero) cause the system to trap at the completion of the faulting instruction. The PC value in the trap record identifies the instruction that would have executed next. The instruction that caused the trap is generally not restartable due to potential modification of its input parameters.

Imprecise faults cannot be attributed to any specific instruction. This is typically because the fault-causing condition is not always detectable during the instruction that caused it, or because the hardware system is having serious problems. In more general terms, an imprecise fault requires that the faulting process be terminated, since it cannot be reliably continued.

The processing method is almost the same for all system faults. As previously discussed, precise faults leave the saved PC pointing to the instruction following the faulting instruction (allowing the possibility for execution to continue), whereas imprecise faults leave the saved PC in a nondeterministic state.

All fault processing uses two control words in CSR: FLTCOD and FLTMSK. Each possible system fault is assigned its own bit in the FLTCOD word. If multiple faults occur simultaneously, multiple bits are set. Each FLTMSK bit corresponds to a bit in FLTCOD. When a FLTMSK bit is set, the corresponding fault is masked (ignored).

When a fault occurs only FLTCOD bits corresponding to current faults, and not masked by FLTMSK, are set; all other FLTCOD bits are cleared.

The fault processing differs from generic trap processing as follows:

1. When a fault occurs, FLTCOD is rewritten: each bit is set if its fault type occurred and was not masked by FLTMSK; all other FLTCOD bits are cleared. Therefore, the fault trap handler sees in the FLTCOD value exactly those fault types causing the current fault trap. The FLTCOD value remains unchanged until another unmasked fault occurs, or until a supervisor-mode csmov instruction explicitly modifies it. Explicit writes to FLTCOD have no effect except to change its value; they do not invoke fault processing.

2. Trap processing is not even initiated if only masked faults occur. If FLTCOD is 0, indicating that only masked faults occurred, no fault trap request occurs. If one or more FLTCOD bits are set, a fault trap request occurs, and trap processing is begun. No bits in the fault mask are changed.

3. All pending PE load and store operations are completed before trap processing begins. This is different from other trap types, which do not wait for pending PE memory operations to complete.

4. The fault trap handler, unlike most other trap handlers, must correctly handle imprecise faults. It must not simply execute crf to return from an imprecise fault, because the faulting program cannot be reliably restarted. (See page 5-11 for the implications of this requirement.)
The fault trap handler can examine FLTCOD to determine which faults caused the current trap, as long as it does not allow another fault trap to occur and change FLTCOD before it has copied FLTCOD’s value. For a floating point (FP) fault, the handler can examine the PPSW to determine which PEs have experienced FP exceptions since the FP exception flags in the PPSW were last cleared. If the handler needs to know which PEs caused the current FP fault trap, it must ensure that the PPSW bits corresponding to all enabled FP exception faults were zero before the faulting instruction was executed. It can do this by clearing these bits as it handles each FP fault trap. The fault trap handler for an FP fault cannot reliably determine which instruction caused the fault, because the FP instructions have different lengths and the PC is advanced past the instruction before the trap occurs. As a result, the handler cannot determine the operation, operands, or result of the trapping instruction.

System Services

The csnall instruction provides a mechanism for a user program to request system services from the kernel. When the instruction is executed, the instruction is stored immediately into TRAPVAL, a CSReg. Then the usual trap processing sequence occurs. The interpretation of the immediate value is not architecturally defined; software conventions are required. This trap is precise and not maskable. It can always be continued.

System Debugging Support

The cbreak (breakpoint) instruction and single-step mode traps provide system debugging support. The cbreak instruction allows a debugger to transfer control from the user program into a trap handler that can process debugger requests. Typically, the debugger program replaces user instructions with cbreak instructions.

Single-step mode provides a mechanism to execute one machine level instruction at a time. The machine mode is controlled by the SS bit in the CPSW, which can only be changed by a supervisor-mode process.

The typical sequence that uses single-step mode starts in the kernel process, debugging user code. The user requests the single-step operation, whereupon the ACU kernel modifies the user’s saved CPSW, enabling the single-step mode. When the csnft instruction is executed, the machine is placed into single-step mode by the saved CPSW being restored. After one user level machine instruction is executed, control is transferred to the single-step trap handler.

For kernel debugging, the only special considerations are the reentrant programming requirements if trying to debug the cbreak trap with breakpoints or single-stepping the single-step trap routine. Except for the breakpoint and single-step handlers, normal
debugging is possible for all kernel routines. Single-stepping a user process does not cause trap handling to single-step, since the single-step mode is always cleared on entry to all trap handlers. Single-stepping a kernel routine usually starts after processing a break-point instruction in that routine, as part of restoring the CPSW with the single-step mode set.

The cbreak and single-step traps are nonmaskable and can always be continued. Further, the trap processing sequence for both is identical to the generic trap processing sequence.

Register Interrupts

The register interrupt table in supervisor CMem provides a bidirectional communication path between the ACU and a VME device. This table consists of four records corresponding to the four asynchronous hardware interrupts, CTRL0A through CTRL0D. The entire table is visible from supervisor CMem and from the VMEbus.

Each record in the register interrupt table includes a control register and seven 32-bit data registers. A VME write to the control register causes a corresponding ACU trap.

A possible use is to allocate one set to communication between the active debugger (on the FE) and the ACU kernel; one to communication between the FE kernel and the ACU kernel; and the remaining two to communication between the ACU and two different IOPs. However, the architecture does not dictate their use.

The specifics of the Register Interrupt Table follow. It contains four records for Interrupt Register sets A, B, C, and D. Each register set contains eight entries with the following descriptions. (The "_" is used to denote one of the Interrupt Register Sets A - D.)

DATA1_ - DATA7_ Data locations DATA1_ through DATA7_ all behave identically. Both VME and CMem reads and writes behave normally. After a new value is stored into a location, reads from either the VME or CMem provides the new value.

CTRL0_ The control register behaves exactly like the data locations, except that it also provides one additional function. Every VMEbus write into this location causes the corresponding bit in the trap request word to be set; this can cause a trap if it is not masked. No other VME or CMem accesses can cause a trap.

This register interrupt architecture allows a variety of uses, including polled bidirectional communication channels, queues, and interrupt driven communication channels. For example, a polled communication channel would follow software conventions in which both the VME device and the ACU would use separate locations and would periodically check for new data using a reply word for acknowledgement. An interrupt interface uses the control location to initiate an ACU interrupt for VME to ACU communication; the ACU would either write a special VME location that would trigger an interrupt on the VME device, or the ACU would use the cirq instruction to generate a VME interrupt.
When using the control registers (CTRL0A - CTRL0D) to generate an ACU interrupt, some software convention must be used to communicate from the ACU to the external device that the interrupt has been serviced. The software convention must preclude a second interrupt request before the first request is acknowledged. This is in contrast to the hardware interrupt acknowledge used in the VMEbus. Two possible software conventions are: always to signal ACU interrupt acknowledge with an interrupt back to the source, or to use a convention on the data stored in the control register. For example, the external device only writes nonzero values into the control register, and, when the ACU acknowledges the interrupt, it writes a zero back into the control register. The external device can only request another interrupt when the control register reads zero.

All register interrupts are precise and maskable and can be always be continued.

**M-Machine (M_Mach) Traps**

The ACU M-Machine is a hardware process that allows PE load/store operations to be overlapped with all other computation (either ACU or PE). All PE registers (PRegs) use a scoreboard interlock system so that PE execution stalls if pending memory operations must be completed for correct operation. When the ACU executes a PE load or store operation, all bounds checking is completed, and the request is queued into the M-Machine. The M-Machine processes all memory requests in a first-in, first-out order. Since the memory operations are occurring asynchronously, all traps are imprecise.

The M-Machine trap provides a mechanism for reporting and handling PE memory access errors, including parity errors and attempts to access protected PMem addresses while in user mode. When a M-machine trap occurs, a status bit is set in the CSREG M-Machine status word, M_STAT. If interrupts are enabled and this interrupt is not masked, usual trap processing occurs. M_STAT can only be cleared by a csmov instruction in a supervisor-mode program.

**Simultaneous Traps**

Due to the asynchronous nature of traps, multiple traps are possible simultaneously. The procedure for handling simultaneous traps was described in the generic trap handling procedure and in the sections describing certain specific trap types. This section provides some additional discussion intended only to make the consequences of that procedure more apparent.
When multiple trap requests are present simultaneously, priority ordering determines the order in which handlers are called, according to Table 5-2.

<table>
<thead>
<tr>
<th>Order</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System Reset—either a hardware reset or cold boot</td>
</tr>
<tr>
<td>1</td>
<td>Faults—either software or hardware faults</td>
</tr>
<tr>
<td>2</td>
<td>ccall—a system call instruction</td>
</tr>
<tr>
<td>3</td>
<td>cbreak—a system breakpoint instruction</td>
</tr>
<tr>
<td>4</td>
<td>Single Step—used by the debugger</td>
</tr>
<tr>
<td>5</td>
<td>All maskable traps (hardware interrupts)</td>
</tr>
</tbody>
</table>

Table 5-2  Simultaneous Trap Processing Priority

For example, a system reset request supersedes all other trap requests and does not allow the system to continue execution, since the PC can be pointing to the middle of an instruction. Pending faults are ignored.

If a fault trap and any lower priority trap occurs, the fault trap is always taken. For imprecise faults, which preclude continuing normal program execution, the fault trap handler generally does not return via crft. The effect of not returning from the fault trap handler with crft is to ignore any pending (threaded) nonmaskable trap requests. In this case, the fault trap handler can choose to enable interrupts (clear ID) and allow them to be handled by their respective trap handlers.

As another example, consider executing a ccall or cbreak instruction while in single-step mode. In either case, the instruction trap has a higher priority than the single-step trap and is taken first. Single-step mode is disabled on trap entry, and the instruction trap handler executes normally. When the instruction trap handler returns using crft, single-step mode is reenabled, and the pending single-step trap is taken. When the single-step trap handler returns using crft, the user program continues execution at the instruction after the ccall or cbreak instruction. The overall effect is that the ccall or cbreak instruction executes like a single instruction from the user program single-step point of view.

In the case of the cbreak instruction, the trap handler provided by the debugger can arrange to have the FE replace the cbreak instruction with the user program instruction originally at that same location; the handler can then explicitly manipulate the saved PC to point to the restored instruction location. The effect is to reexecute from the location that held the cbreak instruction as if the cbreak instruction had never been there.
Chapter 6

System Halts

When the ACU halts, no macrocode-visible state changes. The only ACU state change is in the ACU-VME interface. Specifically, the following VME locations are modified.

- The execution command and status register (ECSR) indicates that the ACU has halted.
- The halt status register (HSR) indicates the reason for the ACU halt.
- The interrupt code register (IRCODE) contains the immediate from the halt instruction.

A VME interrupt can also be generated when the machine halts. (The control of whether a halt causes a VME interrupt is implementation dependent.)

The first maskable trap request (hardware interrupt) that occurs while the ACU is halted is saved until the ACU is restarted (unless the ACU is reset first). This saved maskable trap request is processed normally when the ACU is restarted. However, while the ACU is halted, any additional maskable trap requests beyond the first are ignored and lost.

The ACU halts only for the following reasons:

- A VME device writes the ACU-VME execution and command register (ECSR) telling the ACU to halt. Normally, the current instruction completes, and the machine halts before the next instruction is executed. For two instructions, \texttt{cmov.Tb.Cr} and \texttt{cmov.Cr.Fb}, which can stall the machine indefinitely, a VME halt request aborts the instruction and sets the PC so that the instruction is executed when the ACU is restarted.
- A supervisor routine executes the halt instruction.
With the exception of hardware instability, the machine can continue normal operation after any halt operation caused by a VME device writing to the ACU VME execution command register.
Chapter 7

DPU Initialization (Cold Start)

The DPU initialization sequence is entered only during a power-on sequence or after a low-level machine clear. The following actions occur:

1. Some implementation specific initialization of the internal machine state is performed.

2. Execution begins with a reset trap, as previously described on page 5-7.
7-2  DPU Initialization (Cold Start)
Chapter 8

ACU Interface to the FE

The ACU VME interface is partitioned into four 4-Kbyte pages, pages W, X, Y, and Z, so that each page can be separately mapped into UNIX processes. Selective page mapping provides protection so that only one user has access to the machine resources at a time. The use for each of the four pages follows:

W
All reads and writes into the W page have no effect on the ACU. This page is mapped into all FE user processes that have been swapped out. This mechanism ensures that an inactive FE user process cannot affect an active process running on the DPU; when a process is swapped out and continues writing, it writes to page W, which has no effect.

X
The X page is used by the currently active user process and active debugger. From this page, hardware queues provide user data communication and a set of interrupt registers provide ACU system control for use by the debugger.

Y
The Y page contains mechanisms to directly control and observe the hardware. Only MasPar system routines ever access this page.

Z
The Z page contains interrupt and microcode control registers. Only MasPar system routines ever access this page.

The detailed contents and layout of these pages are implementation specific.
ACU Interface to the FE
Appendix A

Multiuser Operation

This appendix describes the basic method for multiuser operation. A user process includes a UNIX user process on the FE and its associated program on the DPU.

- All user processes access only the DPU X page from the FE.

- The active process, including the active debugger:
  - has the DPU X page mapped into its FE address space and is the only such process.
  - owns all of the ACU user resources.
  - owns all of the PE array user resources.

- The suspended processes, including the inactive debuggers:
  - have the W page of the DPU mapped into the address spaces of the FE in place of the X page, so that a runaway FE process cannot affect another active process.
  - have had their ACU user data removed and stored into supervisor protected memory locations.
  - have had their PE array user data swapped into supervisor-protected high-addressed portions of PE array memory.
• The MasPar instruction memory loader is a privileged process that accesses the DPU Y page and is a shared resource for all user processes. Only other MasPar system programs are allowed to access the Y page (such as the hardware debugger (HDB), system manager, and kernel driver).

• Only the MasPar kernel driver accesses the Z page of the DPU.
Index

Access mode modifiers 2-5
Accumulator 2-21
  flags 3-8
  PE 3-3, 3-29
  specifying 3-20
ACU
  branch and jump instructions 3-14
  calculate instructions 3-14
  condition codes 3-11
Control and Status
  Registers—CSReg 1-5
FE Interface 8-1, 8-2, 8-3, A-1
FE registers 1-6
flags 3-8
halts 6-1
instruction memory 1-5
instructions 2-5
instructions—initial letter 1-10
internal memory 1-4
jump and branch instructions 3-14
memory resources 1-4
operand formats 2-1
processor status word 2-8
register load/store instructions 3-13
  register move instructions 3-12
VME interface 8-1
ACU ↔ PE Move Instructions 3-31
adds_m_ 3-20
Address calculations 3-4, 3-9
Addressing modes 3-4
  uc addressing mode 3-7
  immediate 3-5
Al 3-4
andl_m_ 3-22
ands_m_ 3-21
Array Control Unit (ACU) 1-3
b_c_m_ 3-15
Base 3-4, 3-9
Binary integer instructions
  PE 3-20
Bit numbering 2-2
Block diagram, system 1-1
Boolean instructions
  PE 3-22
bra.Im 3-15
Branch and jump instructions
ACU  3-14
Breakpoints  5-1, 5-2
Broadcast network  1-9
bsr  3-14
bsr.Im  3-15
BUSERR, FLTCOD  2-16
Byte numbering  2-2

C-flag, PPSW  2-23, 3-8
divide by zero  2-17
CO—CReg register  1-5
cadd32_m_  3-14
Calculate instructions
  ACU  3-14
  PE  3-19
cand32_m_  3-14
cbreak  2-11, 5-9, 5-10, 5-12
Ce  3-4
ceq132_m_  3-14
cirq  4-3
cjmp.Cr  3-14, 3-15
cjmp.Ie  3-15
cjsr  3-14
cjsr.Cr  3-15
cjsr.Ie  3-15
cld  1-5
cld32.Ce.Cr  3-13
cld32.Cm.Cr  3-13
clea  3-13
clea32.Ac.Cr  3-13
clea32.Al.Cr  3-13
Cm  3-4
CMem  1-5, 2-4, 2-19, 3-2, 5-10
  accesses  4-2
collection data area  2-21
microcode data area  2-21
regions  2-19, 4-2
Register interrupt area  2-21
Supervisor data area  2-21
Supervisor region  2-20
trap table  2-21
user region  2-19
VME region  2-20
CMem address
  outside limit  2-13
CMem regions  1-5
CMem—ACU memory  2-19
CMEMALIGN, FLTCOD  2-15
CMEMLIM  2-6
CMEMLIM—CMem Limit Register
  2-9
CMEMLIMIT, FLTCOD  2-13
cmov.Cr.Fb  1-6, 3-12
cmov.Tb.Cr  1-6, 3-12
cmov32_m_  3-12, 3-12
Communication flow  3-26
Communication status flags - PE
  3-26
Communications
  instructions: PE-to-PE  3-25
Concurrent execution  1-7
Condition code move instructions
  PE  3-17
Condition codes  3-11
Contingent execution mechanism  1-8
Control flow  1-9
Control register  5-10
cor32_m_  3-14
CPSW  2-6, 4-4, 5-5, 5-6, 5-7, 5-10
  flags  2-8
  updating with csmov  4-4
CPSW—ACU Processor Status Word
  2-8
Cr  3-4
CReg  1-5, 2-4, 3-2
  layout  2-6
CReg—ACU general purpose
  registers  2-6
CReg[... ]  3-2
crft  4-3, 5-4, 5-6, 5-8, 5-9, 5-12
cscall  2-11, 4-4, 5-9, 5-12
cshl32_m_  3-14
cshra32_m_  3-14
cshrl32_m_  3-14
csmov  4-4, 5-8
cmov32_m_  3-12
CSReg  1-5, 2-4, 3-2, 5-9, 5-11
CSReg—ACU special registers  2-6
CSReg[... ]  3-2
cst  1-5
cst32.Cr.Ce  3-13
cst32.Cr.Cm  3-13
csub32_m_  3-14
csubr32_m_  3-14
CTLACUPARITY, FLTCOD  2-13
CTLIOPARITY, FLTCOD  2-13
CTLPEPARITY, FLTCOD  2-13
CTRL0A  5-10
CTRL0D  5-10
CTRL0_  5-10
cxor32_m_  3-14

Data types  1-7
DATA1_  5-10
Direct-mapped VME space  2-20
disable 3-33
disable.Cr 3-33
disable.Ie 3-33
disable.Im 3-33
dist 3-32
dist1.Cr.Pe 3-32
dist1.Cr.Pr 3-32
dist2.Cr.Pr 3-32
DIV0, FLTCOD 2-16
div_s_m_ 3-20
divu_s_m_ 3-20
DPU
   initialization 7-1
   instruction description format 3-7
   instruction formats 3-5
   instruction set 1-10
E-bit 3-20, 3-33
E-bit contingent execution 3-7
E-bit, PPSW 2-23, 3-8
ECC_HARD, PPSW 2-26
ECC_SOFT, PPSW 2-27
ECR 5-7, 6-1
Effective address 3-4, 3-9
elseluc_s_m_ 3-23
enable 3-33
enable.Cr 3-33
enable.Ie 3-33
enable.Im 3-33
eq 3-11
Execution operations 1-8
f conversion type 3-24
F-flag, PPSW 2-24, 3-8, 3-27
fadds_s_m_ 3-23
Fault processing 5-8
Faust 5-7
   imprecise 5-8
   precise 5-8
Fb 3-3, 3-4
fceil_s_f_s_s 3-25
FD, PPSW 2-17
FD-flag, PPSW 2-25, 3-8
FDIV0, FLTCOD 2-17
fddiv_s_m_ 3-23
FE 1-3
FE kernel 5-10
FE registers
   ACU 1-6
   FE-DPU Interface 2-5
fegls_s_m_ 3-23
ffloor_s_f_s_s 3-25
FI, PPSW 2-17
FI-flag, PPSW 2-25, 3-8
FINEXACT, FLTCOD 2-17
FINVOP, FLTCOD 2-17
Flag move instructions
   PE 3-17
Flag
   accumulator 3-8
   ACU 3-8
   communication status: PE 3-26
flg 3-3
Floating-point formats 2-2
   IEEE 2-2
   VAX 2-2
Floating-point instructions
   binary - PE 3-23
   conversion - PE 3-24
   unary - PE 3-23
Floating-point operands 2-1, 2-2
Floating-point operations 1-8
Floating-point representations 2-2
Floating-point semantics 2-2
FLTCOD 2-6, 2-17, 5-8
   FP fault trap handler 5-9
FLTCOD—Fault Code Word 2-11
FLTMSK 2-6, 5-8
FLTMSK—Software Fault Mask 2-17
fmul_s_m_ 3-23
fneg_s_.Pr.Ps 3-24
FOVFLO, FLTCOD 2-17
FRBEQ 1-6, 3-12
fmod_s_m_ 3-23
Front End (see also FE) 1-3
frounds_s_f_s_f 3-25
frounds_s_f_s_s 3-25
frounds_s_g_s_f 3-25
fsqrt_s_.Pr.Ps 3-24
fsub_s_m_ 3-23
fsubr_s_m_ 3-23
ftruncs_s_f_s_s 3-25
ftruncu_s_.Pe 3-24
FU, PPSW 2-17
FU-flag, PPSW 2-25, 3-8
FUNFLO, FLTCOD 2-17
FV, PPSW 2-17
FV-flag, PPSW 2-25, 3-8
FX, PPSW 2-17
FX-flag, PPSW 2-25, 3-8
Global router network 3-25
gor 3-32
gorl.Pc.Cr 3-32
gorl.Pc.Cr 3-32
gor.s_.Pr.Cr 3-32
graddr 3-33
graddr.Pe 3-33
gt 3-11

halt 4-3
Halt 6-1
Hardware interrupts 5-1, 5-2
hi 3-11
hs 3-11
HSR 6-1

I/O 4-4
I/O devices and mappings 3-31
I/O Instructions - PE 3-31
I/O RAM
  direct access - user mode 4-4
I/O table 4-4
Ib 3-4, 3-5
  .Ib.Pe(5) 3-16
Ie 3-4, 3-5
  .Ie.Pe(4) 3-16
IEEE floating-point 2-2
Illegal accesses 2-5
ILLOP, FLTCOD 2-15
ILLPEADDR, FLTCOD 2-16
Im 3-4, 3-5
IMem 2-4, 2-5
  accesses 4-3
  address space 2-5
IMem—ACU Instruction Memory 1-5, 2-5
IMEMALIGN, FLTCOD 2-15
IMEMPARITY, FLTCOD 2-14
Immediate addressing modes 3-5
Immediate fields
  formats 3-5
Immediates
  sign extension 3-4
Imprecise faults 5-8
Instruction description format
  DPU 3-7
Instruction formats, DPU 3-5
Instruction memory 1-5
Instruction memory page map 1-5
Instruction memory, ACU 1-5
Instructions
  effect of mode 4-3
  supervisor-only 4-3
Integer conversion instructions
  PE 3-21

Integer instructions
  PE 3-20
IOERR, FLTCOD 2-15
IOINDEX (CS18) 3-31
IOINDEX 2-6
IOINDEX—Index Into I/O Table 2-18
IORD 4-4
iords_ 3-31
IOTBLBASE 2-6, 4-4
IOTBLBASE—I/O Table Base 2-18
IOTBLSZ 2-6, 4-4
IOTBLSZ—I/O Table Size 2-19
IOWRT 4-4
iowrts_ 3-31
IRCODE 6-1
Is 3-4, 3-5
  .Is.Pr(2) 3-16

JMP_ 5-5, 5-6
Jump and branch instructions
  ACU 3-14

Kernel
  ACU 5-10
  FE 5-10
Kernel debugging 5-9
Kernel driver A-2
Kernel services 5-9
  requesting 5-9

L-bit, PPSW 2-23, 3-8, 3-22
ld and ECC_HARD 2-26
ld and ECC_SOFT 2-27
lds.s_.Pm.Pe 3-18
lds.s_.Pz.Pe 3-18
lds.suc.Pm.Pe 3-18
lds.suc.Px.Pe 3-18
ldsol and ECC_HARD 2-26
ldsol and ECC_SOFT 2-27
ldsol 3-19
ldsol.s_.Pm.Pe 3-18
ldsol.s_.Pz.Pe 3-18
ldsol_.Pz.Pe and PMEMLIMIT 2-27
ld_.Px.Pe and PMEMLIMIT 2-27
ld_.Px.Pe 3-19
lds_/st_s_instructions 3-19
ld_nf and ECC_HARD 2-26
ld_nf and ECC_SOFT 2-27
lds.uc.Px.Pe and PMEMLIMIT 2-27
ld uc and ECC_HARD 2-26
ld uc and ECC_SOFT 2-27
ld uc.Px.Pe and PMEMLIMIT 2-27
ld uc/st uc instructions 3-19
le 3-11
lo 3-11
Load and store operations 1-7
Load operations
PE, E- and M-bits 3-19
Load/store error 2-10
Load/store instructions
register: ACU 3-13
ls 3-11
LSB, PPSW 2-23
LSB, PPSW accumulator 3-8
lt 3-11
M-bit, PPSW 1-8, 2-24, 3-8
M-flag 3-19
M-Machine
interrupts 5-11
processing 5-11
traps 5-11
MACHFAULT, FLTCOD 2-15
Memory map 1-5
Memory operations 1-8
mi 3-11
mod_s_m_ 3-20
Modes
effect on instructions 4-3
modu_s_m_ 3-20
mov 3-29
mov.Cr.Fb 5-5
mov.Tb.Cr 5-5
movlbm_ 3-17
movluc.Im.Pr 3-7
movluo_m_ 3-17
mov32_m_ 3-16
Move instructions
condition code: PE 3-17
flag: PE 3-17
register: ACU 3-12
register: PE 3-16
special register: PE 3-17
MP-1 subsystems 1-2
mul_s_m_ 3-9, 3-20
mulus_m_m_ 3-20
M_Mach 5-11
N-flag, PPSW 2-23, 3-8
ne 3-11
Network
Broadcast 1-9
Global router 1-9
OR-reduction 1-9
X-Net 1-9
No 3-4
nop, No 3-33
Notation 3-9
Numbering
PE 2-3, 3-33
Offset 3-4, 3-9
Offset field
PReg 3-8
Operand formats 2-1
Operand overlap 3-8
Operands
floating-point 2-1, 2-2
formats 2-1
little-endian 2-2
Operations
endian-ness 1-5
orlm_m_ 3-22
ors_m_m_ 3-21
Overlap
operands 3-8
Page W 8-1
Page X 8-1
Page Y 8-1
Page Z 8-1
Parallel PE I/O 4-4
Parity error
ACU program memory 2-14
ACU-control bits 2-13
I/O-control bits 2-13
PE-control bits 2-13
Router instruction 2-14
X-Nnet instruction 2-14
pbcl_.Pe 3-18
PC 5-5, 5-6
PC offsets 3-15
PC—program counter 2-4
pconv_s_s_s 3-22
pconv_s_s_u 3-22
pconv_u_s_s 3-22
pconv_u_u_u 3-22
PC_ 5-4
Pd 3-4
.Pd.Pr(2) 3-16
Pe 3-4
PE ACU Move Instructions 3-31
PE
accumulator 2-22, 3-3, 3-29
accumulator LSB 2-23
address fields 2-3
binary integer instructions 3-20
boolean instructions 3-22
calculate instructions 3-19
carry flag 2-23
communication status flags 3-26
communications instructions 3-25
condition code move instructions 3-17
condition codes 3-11
E-bit 3-20
enable bit 2-23
F-flag 3-27
flag move instructions 3-17
floating-point binary instructions 3-23
floating-point conversion instructions 3-24
floating-point divide by zero flag 2-25
floating-point inexact flag 2-25
floating-point invalid flag 2-25
floating-point overflow flag 2-25
floating-point unary instructions 3-23
floating-point underflow flag 2-25
general purpose registers 2-22
instructions—The Hardware Programming Manual contains the exact VME and CMem address of each set
integer conversion instructions 3-21
integer instructions 3-20
L-bit 3-22
logical accumulator 2-23
memory 1-6
memory access bit 2-24
negative flag 2-23
numbering 2-3, 3-33
operand formats 2-1
overflow flag 2-23
PMem address limit error 2-27
PMem corrected (soft) ECC error 2-27
PMem uncorrectable (hard) ECC error 2-26
processor status word 2-22
R-flag 3-26, 3-29
register move instructions 3-16
register operations 1-6
router fetch flag 2-24
router parity error 2-26
router receive flag 2-24
router transmit flag 2-24
special purpose registers 2-22
special register move instructions, PE 3-17
T-flag 3-25, 3-26
unary integer instructions 3-21
X-network parity error 2-26
zero flag 2-23
PE accumulator 2-28
PE array 1-6
PE array edges
interconnection 2-4
PE flag addresses 3-8
PE general purpose registers 2-22
PE I/O Instructions 3-31
PE numbering 2-3, 3-33
PE special purpose registers 2-22
.PE.PE(7) 3-16
pl 3-11
Pm 3-4
PMem 1-6, 2-22, 3-3, 3-12
access 1-7
accesses 4-3
regions 4-3
supervisor region 2-28
user region 2-28
PMem address
outside limit 2-13
PMem—PE memory 2-28
PMEMLIM 2-6
PMEMLIM—PMem Limit Register 2-9
PMEMLIMIT 4-3
PMEMLIMIT, FLTCode 2-13
PMEMLIMIT, PPSW 2-27
PMSTAT 2-10
PMSTAT—PMem Status Word 2-10
PPSW 2-14, 3-8
flags 2-22
FP fault trap handler 5-9
PPSW—PE processor status word 2-10, 2-22
Pr 3-4
.Pr.Pd(2) 3-16
.Pr.Ps(2) 3-16
Precise faults 5-8
PReg address
outside limit 2-13
PReg  1-6, 2-21, 3-3, 3-33
   access  1-7
   offset  3-20
   offset field  3-8
PReg—PE general purpose registers 2-22
PREGALIGN, FLTCOD 2-15
PREGLIM 2-6
PREGLIM—PReg Limit Register 2-9
PREGLIMIT 4-3
PREGLIMIT, FLTCOD 2-13
PRegs 5-11
PReg[...] 3-3
Processor numbering 2-3, 3-33
Program Counter (PC) 2-19
Programming model 1-9
Ps 3-4
PSW_ 5-4
Px 3-4
Px mode 3-4
Pz 3-4
QSR 2-6
QSR—Queue Status Register 2-8
Queues 1-6
R-flag, PPSW 2-24, 3-8, 3-26, 3-29, 3-30
rcfetch 3-26
rclose 3-26
rclose.No 3-27
Register interrupt architecture 5-10
Register interrupt table 5-10
Register interrupts 5-11
Register load/store instructions
   ACU 3-13
Register move instructions
   ACU 3-12
   PE 3-16
Registers
   ACU general purpose 2-6
   ACU program registers—CReg 1-5
   ACU special 2-6
   CO—CReg 1-5
   CReg 1-5
   PE 1-6
Reset trap 5-7
rfetch 3-26
rfetchc_s_.Pe.Pe 3-27
rfetchc_s_.Pe.Pe 3-27
ropen 3-26
ropen.Px 3-27
rosend 3-26
rosend_s_.Pe.Pe.Pe 3-27
Router 1-9
Router addresses 2-3, 3-33
Router instructions 3-26
Router network 3-25
Row-column format 2-3
RPROTO, FLTCOD 2-16
rsend 3-26
rsend_s_.Pe.Pe 3-27
RTRPARITY, FLTCOD 2-14
RTRTNOR, FLTCOD 2-14
RTR_ERR, PPSW 2-26
RUNENBL 5-7
s conversion type 3-21, 3-24
Scratch register 2-21
shla_s_m_ 3-21
shll_s_m_ 3-21
shra_s_m_ 3-21
shrl_s_m_ 3-21
Sign extension
   immediates 3-4
SIMD 1-6
Simultaneous traps 5-11
Single step events 5-1, 5-2
Single-step mode 5-9
smov 2-27
smov32 2-27
smov32_m_ 3-17
Soft PMem ECC 2-10
SOLERR 2-10
Solitary load/store error 2-10
Special register move instructions
   PE 3-17
SPMEMECC 2-10
Sr 3-4
SReg 2-21
SReg—PE special purpose registers 2-22
SReg[...] 3-3
st_s_.Pe.Pm 3-18
st_s_.Pe.Px 3-18
st_s_.uc.Pe.Pm 3-18
st_s_.uc.Pe.Px 3-18
Store operations
   PE, E- and M-bits 3-19
stsol and ECC_HARD 2-26
stsol and ECC_SOFT 2-27
stsol 3-19
stsol_s_.Pe.Pm 3-18
Index

\texttt{stsol_..Pe.Px} 3-19
\texttt{stsol_..Pe.Px and PMEMLIMIT 2-27}
\texttt{st_..Pe.Px and PMEMLIMIT 2-27}
\texttt{st_..Pe.Px 3-19}
\texttt{st..uc and ECC_HARD 2-26}
\texttt{st..uc and ECC_SOFT 2-27}
\texttt{st..uc.Pe.Px and PMEMLIMIT 2-27}
\texttt{sub_s_m} 3-9, 3-20
\texttt{subrs_m} 3-9, 3-20
Supervisor mode 4-1
Supervisor region
  CMem 2-20
  PMem 2-28
Supervisor region, CMem 1-5
Supervisor-mode bit
  changing 4-4
  Supervisor-only instructions 4-3
  Suspended processes
    multiuser operation A-1
SWAP 2-6
SWAP—SWAP Register 2-19
System calls 5-1, 5-2
System debugging support 5-9
System faults 5-1, 5-2
T-flag, PPSW 2-24, 3-8, 3-25, 3-26, 3-28, 3-29, 3-30
Tb 3-3, 3-4
Threaded trap requests 5-6
TOBEQ 1-6, 3-12
Too many soft PMem ECC 2-10
Toroidal wrap 2-4
TPMEEMCC 2-10
Trap 5-6
Trap control record 5-2
Trap parameter value 2-11
Trap table 5-2
  assignments 5-4
  JMP_locations 5-6
TRAPMSK 2-6, 5-3, 5-5
TRAPMSK—Trap Mask Word 2-18
TRAPREQ 2-6, 5-3, 5-5, 5-6
TRAPREQ—Trap Request Word 2-17
Traps 5-1
  disabling 5-6
  precise 5-2
  processing 5-5, 5-6
  simultaneous 5-11
  threaded requests 5-6
TRAPVAL 2-6, 5-9
TRAPVAL—Trap Parameter Value 2-11
tst_s_m 3-21
u conversion type 3-21
uc 3-17
Unary integer instructions
  PE 3-21
Uncorrectable PMem ECC 2-10
UPMEEMCC 2-10
User region
  CMem 2-19
  PMem 2-28
User region, CMem 1-5
V-flag, PPSW 2-23, 3-8
VAX byte numbering format 2-2
VAX floating-point format 2-2
vc 3-11
VME devices
  direct access - user mode 4-4
VME space 2-20
vs 3-11
W page 8-1
X page 8-1
X-Network 1-9, 3-25
  examples 3-30
  instructions 3-27, 3-29
  instructions - nonpipelined 3-31
  instructions - pipelined 3-30
xnet 3-29
Xnet 1-9
xnetc 3-29
xnetp 3-29
XNETPARITY, FLTCD 2-14
xnetpE8.Pe.Pe.Cr 3-31
xnetE8.Pe.Pe.Cr 3-30
xnet_d_s_m 3-28
xnetc_d_s_m 3-28
xnetp_d_s_m 3-28
XNET_ERR bit, PPSW 2-14
XNET_ERR, PPSW 2-26
xor1_m 3-22
xor_s_m 3-21
Y page 8-1, A-2
Z page 8-1, A-2
Z-flag, PPSW 2-23, 3-8
zero_s_m 3-21