CONVEX Troubleshooting Guide (C201, C202, C210, C220)

Document No. 081-000930-201

First Edition, Rev. 1
February 1989

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX Troubleshooting Guide
(C201, C202, C210, C220)
Order No. DHW-098
First Edition, Rev. 1

Printed in the United States of America

Revision Sheet

CONVEX Troubleshooting Guide
(C201, C202, C210, C220)

Edition	Document No.	Date	Description
First	081-000930-200	September 1988	First release.
First, Rev. 1	081-000930-201	February 1989	Corrected errors, added new material to Chapters 6–10, and updated appendixes.

FCC NOTICE

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in strict accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Do not connect external equipment to the utility outlets in Convex equipment cabinets. Unauthorized connection voids all agencies' emissions certifications.

Table of Contents

7 Hard Error Messages

8 I/O Error Messages

Appendixes

List of Tables

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

Purpose and Intended Audience

The *CONVEX Troubleshooting Guide* is the fifth of six volumes in the *CONVEX Maintenance Documentation (C201, C202, C210, C220).* The other volumes include the following:

- *CONVEX Maintenance Documentation Overview (C201, C202, C210, C220)*
- *CONVEX Theory of Operation (C201, C202, C210, C220)*
- *CONVEX Installation Guide (C201, C202, C210, C220)*
- *CONVEX General Maintenance Guide (C201, C202, C210, C220)*
- *CONVEX Removal/Replacement and IPB Guide (C201, C202, C210, C220)*

The main purpose of this document is to assist the Field Engineer (FE) in getting the customer's machine up and running as quickly and easily as possible. To that end, background information, troubleshooting procedures, and removal-replacement procedures to assist the FE in repairing the customer's machine in 80% to 90% of service calls is provided. Additionally, this document may be used as a teaching text by the CONVEX Training department.

Scope

The material in this volume applies to CONVEX C201, C202, C210, and C220 supercomputers.

Outline

The content of each chapter is outlined below:

Chapter 1. Introduction—This chapter contains the philosophy and methodology of troubleshooting used in the guide

Chapter 2. System Dead—This chapter provides information about the processor cabinet AC and DC power systems and how to troubleshoot a system that will not power up

Chapter 3. Fails During Boot—This chapter contains information detailing the boot process and offers guidance on how the troubleshoot a system that fails during booting

Chapter 4. Fails During UNIX Process Execution—This chapter is devoted to troubleshooting the system that fails during the execution of a program under UNIX

Chapter 5. Fails Diagnostics—This chapter provides information on how the diagnostic tests work, their limitations, and how to troubleshoot a machine that fails diagnostic tests

Chapter 6. Indicators—This chapter details the various indicators, explains the operation of the System Control Module (SCM) and its purpose in the system, and lists the **SYSTEM STATUS** error codes with explanations

Chapter 7. Hard Error Messages—This chapter lists the various hard error messages and provides information on what the error means, where it was detected, and boards that are likely suspects

Chapter 8. I/O Error Messages—This chapter lists messages that might be displayed by CCU EGOS or one of the various I/O drivers, and provides a definition as well as a cause and disposition for each

Chapter 9. SPU UNIX Messages—This chapter lists messages that might be displayed by SPU UNIX and provides a definition as well as a cause and disposition for each

Chapter 10. CONVEX UNIX Messages—This chapter contains a list of messages that might be displayed by CONVEX UNIX and provides a definition as well as a cause and disposition for each

Appendix A. Crash Dump Procedure—This appendix provides step-by-step instructions detailing how to obtain a crash dump that can be sent to CONVEX for analysis

Appendix B. CPU Instruction Glossary—This appendix contains a list of CPU instruction mnemonics with a description of each one, and a list of boards to suspect as likely cause when the instruction fails

Appendix C. ASP Entry Point Listing—This appendix contains ASP entry points (listed in tables by both entry point and opcode) as an aid in troubleshooting a system hang or halt situation

Appendix D. Wire Lists—This appendix provides wire lists, routing, and connector pin definitions for system cables

Appendix E. Problem Reporting—This appendix contains information about using the *contact* facility to report problems

Notational Conventions

The notational conventions used in this text are listed below:

- When directed to *enter* a particular command, pressing *Return* after the command has been typed is implied
- When directed to *type* a particular string, pressing *return* is *NOT* implied—the desired action will be initiated by the string (control key sequences, for example)
- A *register* is a programmer-visible hardware storage element internal to the processor
- *Main memory* or *physical memory* is the physical storage installed in the processor
- *Logical memory* or *virtual memory* is the perceived amount of main memory as seen by the application programmer

- The symbol *K* is an abbreviation for *kilo* or 1,024
- The symbol *M* is an abbreviation for *mega* or 1,048,576
- TBD is an abbreviation for *To Be Determined*

The following are examples of warnings, cautions, and notes and their typical content and location as used in CONVEX documents:

WARNING

Warnings highlight procedures or information necessary to avoid injury to personnel. A warning immediately precedes the critical information and includes a description of the hazard.

CAUTION

Cautions highlight procedures or information necessary to avoid damage to equipment, loss of data, or invalid test results. A caution immediately precedes the critical information and includes a description of the possible damage.

NOTE

A note highlights useful information that is supplemental in nature. A note may immediately precede or follow the information that is being highlighted.

Associated Documents

The following is a partial list of other manuals or books that may provide more detailed information on the topics presented in this manual:

- *CONVEX Diagnostics Documentation (C200 Series)*, Product No. DHW-080
- *CONVEX Processor Operation Guide*, Product No. DHW-015

Ordering Documentation

To order the most current version of this or any other CONVEX document, use the CONVEX product number. If the product number is not known, order by the exact title. In some situations, the most current version may not be desired. To receive a specific version of a manual, order the manual by its document, or part, number, which can be obtained by contacting the local CONVEX office or by calling the Technical Assistance Center.

The product number for this manual is DHW-098.

The document number for this manual is 081-000930-201.

CONVEX documents can be ordered by mail by sending a request to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

Technical Assistance

Hardware and software support can be obtained through the CONVEX Technical Assistance Center (TAC). In Texas, the TAC can be reached by calling (214)952-0379. From other locations in the continental United States, call 1(800)952-0379. Customers outside the United States should contact their local CONVEX office.

Electronic Mail

The Hardware Documentation Group has an email address for documentation comments. Use this service to give us a quick response mechanism if you have special documentation questions that you would like addressed immediately. If you have a technical question, you should still contact the Technical Assistance Center, as described previously. To use email response service, just send mail addressed to:

`cnvxhwdoc@convex.COM`

We will read your comments and give you a personal reply.

What to Include in an Email Message

When you use the electronic mail service, please provide the following information:

- The reader's name and company name
- A return email address in INTERNET notation or UUCP (bang) notation
- The manual that is being critiqued
- The chapter and page number in question
- The comment

Reader's Forum

If you wish to mail your comments to us, please use the form at the end of this manual and list the document page number with your questions and comments. Thank you.

Acknowledgments

I would like to thank the following people for their contributions to this document:

- Technical contributors: Rich Adkisson, Tony Brewer, Art Clark, Mary Cooley, Brad Culter, Gary Gostin, Carl Jackson, Jeff Jones, Tony Jones, Art Kimmel, Brian Konigsburg, Kevin Lowderman, Craig Reed, and Marc Quattromani
- Review team members: Art Clark, Don Davis, Ron Engelking, and Art Kimmel
- Hardware Documentation staff: Larry Bonura, Art Fischman, and Jimmie Holman

Without the efforts of all the aforementioned, this document would not have been possible.

Leigh Ellert, Lead Writer
CONVEX Hardware Documentation

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1
Introduction

1.1 Overview

This chapter details the CONVEX C200 Series troubleshooting philosophy and discusses the troubleshooting methodology. A table outlining failure modes is also provided.

1.2 CONVEX C200 Series Troubleshooting Philosophy

The objective of the CONVEX Field Organization is to return the customer's machine to service as quickly as possible.

All the elements of the C200 Series maintenance philosophy have been driven by that goal. The selection of the Field Replaceable Units (FRUs), for example, not only has a large impact on the required level of FE training and sparing, but also determines the amount of time that will be needed to remove and replace a failed part.

How long it takes to remove and replace a failed part is one term of the equation that describes how long it will take to return the customer's machine to service.

The other term of the equation is how long it takes to determine which part has failed so that it can be replaced. This, however, is largely dependent upon what tools are available to troubleshoot the machine and isolate the failure to an FRU that can be replaced. This portion of the equation typically consumes about 90% of the time required to repair the machine.

The C200 Series troubleshooting tools include the following:

- **This guide** which contains information, definitions, procedures, and guidelines for troubleshooting the system
- **Indicators** by which the system is able to communicate certain information about its status
- **Error messages** that might be generated as the machine failed
- **Diagnostic tests** that verify that the various functions in the machine operate as they were designed
- **Field Engineer knowledge** of the system architecture, how the machine operates, how the diagnostic tests work and what their strong points and limitations are, and experience with the system

Unfortunately, on a system of this size and complexity, it is very difficult to implement diagnostic tests that are 100% comprehensive and accurate, and that correctly specify the FRU that is defective and should be replaced.

Then there are times when a machine fails while running an application program, and the diagnostic tests do not fail.

This is where the FE's knowledge and skill comes in.

1.3 C2 Troubleshooting Methodology

When a service call is received, there will always be some sort of a failure symptom that prompted the customer to call.

When the FE arrives at the site, however, the original symptoms may not be present. The customer may have removed power from the system or taken some other action that resulted in no symptoms or new symptoms that are different from the original symptoms. TAC personnel may also have tried to troubleshoot the problem over the phone while the FE was en route to the site.

If the original symptoms have been disturbed as a result of remote TAC troubleshooting or customer activity, the FE will need to know the initial symptoms to evaluate the situation and decide to how to troubleshoot the machine.

Once the initial failure symptoms have been established, the FE can use the failure symptoms along with knowledge of the machine architecture, how it operates, and what it was doing when it failed, to determine the most likely failure mode.

The possible failure conditions have been grouped into four very broad categories:

- **System Dead**—System will not power-up
- **System Fails During Boot Process**—System will not boot all the way up to CONVEX UNIX
- **System Fails During Execution of Job**—System suffers failure during execution of UNIX process or application
- **System Fails Diagnostics**—System fails one or more diagnostic tests

Each of these broad categories is then divided into several narrower failure modes.

The following table shows these relationships:

Table 1–1, Failure Modes

Category	Failure Mode	Likely Cause	Disposition
System Dead	AC power bad	Site/cabinet breakers, input power filter	Check AC power, path
	P/S breaker tripped	Failed power supply	Replace power supply
	SCM shutdown	Environmental envelope exceeded	correct and power up
System Fails During Boot Process	SPU fails POST	SP2, No clocks	Check SPU, SCM, PIA
	SPU boot fails	SP2, SPU disk, SCSI	Check SP2, SPU disk
	CPU init fails	SP2, CPU functions	Run *spu4000*, change board
	CPU boot fails	SP2, CPU functions	Run Diagnostics
System Fails During Execution of Job	Hang	Missed transaction	Reboot, run diagnostics
	Crash — UNIX	Kernel lost	Reboot, rerun job
	Crash — Hard error	Invalid data detected	Rerun job, run diagnostics
	Wrong answer	Invalid code, design problem	Rerun job, run diagnostics
	WA/core dump	Illegal operation	Rerun job
System Fails Diagnostic	SPU diagnostic	Failed SP2	Replace SP2
	Memory diagnostics	Failed memory board	Replace memory board
	CPU diagnostics	Microcode/boards	Replace code/boards
	I/O diagnostics	Failed I/O board	Replace board

Once the failure mode has been determined, careful consideration of the initial failure symptoms, the current evidence (indicators, error messages, error log files, etc.), along with knowledge of how the machine operates and what it was doing at the time of the failure, should suggest a particular area of the machine that is probably to blame.

When an area of the machine has been identified as the probable cause of the trouble, diagnostic tests can be run on that area to attempt to verify and localize the failure.

Then, based on the diagnostic tests that fail, the suspected board, boards, or peripherals can be replaced and the failing diagnostic tests run again to verify that the problem was corrected by the replacement.

When diagnostic tests, other software testing tools (*sysex*, etc.), and the application program no longer fail, the machine is considered repaired.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2
System Dead

2.1 Overview

This chapter provides information about the processor cabinet AC and DC power systems and guidelines for troubleshooting a system that will not power up.

2.2 AC Power Bad

This section discusses the problems that can result in AC power not being available to the DC power supply circuit breakers.

TBD

2.3 Power Supply Circuit Breaker Tripped

This section discusses the problems that can result in the tripping of a power supply circuit breaker.

TBD

2.4 SCM Shut Down

This section discusses the problems that can result in the System Control Module shutting down the system.

TBD

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3
Fails During Boot

3.1 Overview

This chapter details the booting process through each of the three transitions between power **OFF** and CONVEX UNIX. First, each of the transitions is explained. Then, what is likely to go wrong during the transition is discussed. Finally, guidelines are offered for troubleshooting a machine that fails during a transition.

3.2 Introduction

System **OFF** to the CONVEX UNIX prompt involves three major transitions:

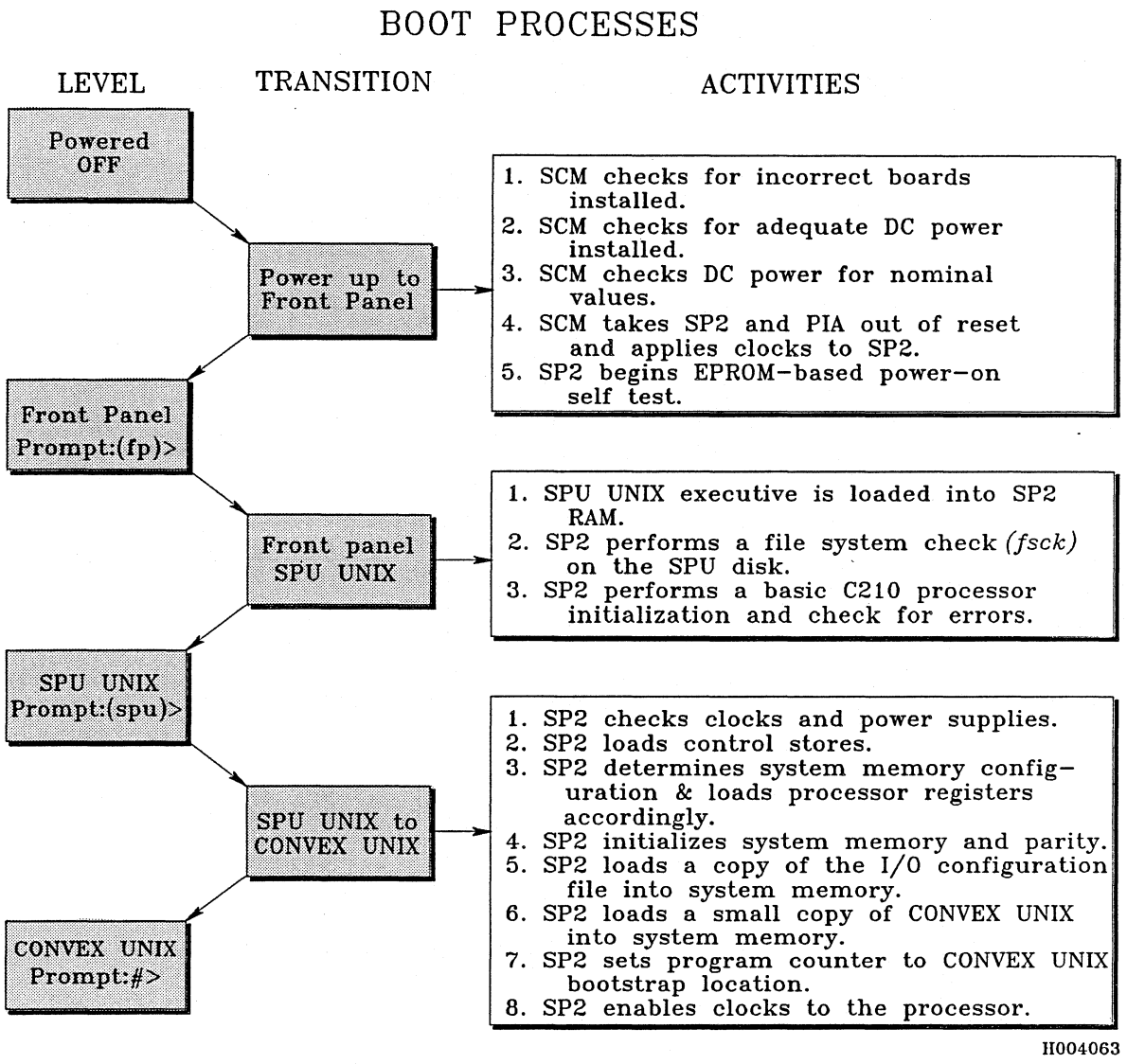
- **Power up to SPU front panel**—The System Control Module (SCM) performs system pre-powerup activities and the SP2 performs Power-On Self-Test (POST)
- **SPU front panel to SPU UNIX**—SPU UNIX is loaded from the SPU boot device and then executed
- **SPU UNIX to CONVEX UNIX**—SP2 initializes CPU and system main memory, loads control stores, and enables clocks to the CPU

These transitions separate four levels:

- **Power OFF**—AC power to the DC power supplies not enabled, only the SCM is powered up—this is an inactive level
- **SPU front panel** (`(fp)>`) —A firmware-resident monitor, allowing user access to many SPU functions without SPU UNIX being loaded
- **SPU UNIX** (`(spu)>`) —The SP2 operating system
- **CONVEX UNIX** (`#>`) —The CPU operating system

The following figure illustrates the relationship between the different levels and the transitions between them:

Figure 3–1, Boot Processes



The system can suffer a failure while operating at any level, or during a transition between levels.

This chapter discusses the transitions between levels, what failures are likely, and offers guidance on how to troubleshoot these problems. Failures during execution of jobs under UNIX are covered in Chapter 4, "Fails During UNIX Process Execution."

3.3 C200 Boot Processes

The SPU performs several tasks in support of the C200 boot process. First, it performs an extensive Power-On Self-Test (POST) to ensure that all internal subsystems and functions are working correctly. Second, it boots its own version of UNIX which allows it to function as an independent computer in its role of booting and supporting the CONVEX C200 processor. Third, the SPU initializes the CPU (or CPUs) to the proper state so that the CPU can be booted. Finally, the SPU boots CONVEX UNIX on the CPU.

There can be some variation in the CPU boot process. This is especially true in the areas of the quantity and detail level of the boot progress reports from the SPU, the degree of user intervention required to complete the boot process, and when and where the entry into the automatic boot procedure is initiated.

The boot process encompasses a wide spectrum. At one extreme, the operator turns the front panel keyswitch from **0 OFF** to the **1 SECURE EXECUTION** position and waits for the entire boot process to automatically complete. At the other extreme, the operator initiates each step from the local or remote console after the front panel keyswitch is set to the **1 LOCAL MAINTENANCE** or **1 REMOTE MAINTENANCE** position. The operator can also initiate some number of steps and then issue an automatic boot order from that point.

The secure execution boot process requires no operator intervention and generates the fewest and simplest reports on the system console or printer. The largest, most detailed reports result when the operator manually initiates each step of the boot process. Boot procedures between these extremes generate detailed reports up to the point where the automatic boot process is initiated.

Regardless of the boot method, each variation of the C200 boot process requires the services of the SPU and includes the following steps, whether operator-initiated or part of the secure execution boot process:

- The operator turns the front panel keyswitch from **0 OFF** to the **1 LOCAL MAINTENANCE**, **1 REMOTE MAINTENANCE** or **1 SECURE EXECUTION** position.
- System Control Module (SCM) examination and approval of system configuration and power
- Successful SP2 power-up, EPROM-based self-testing, and entry into front panel mode
- Booting the SP2 with SPU UNIX
- Initialization of the C200 processor(s)
- Booting the processor with CONVEX UNIX

3.4 Power OFF to Front Panel

After the SCM determines that it is safe to start the machine, DC is applied to the power supply buses, and the PIA/SP2 reset is released.

Next, the SPU begins its built-in POST.

A failure can occur during either of these processes, and will most likely be caused by a problem on one or more of the following boards.

- SCM
- PIA
- SPU

3.4.1 System Power Up

Once the front panel keyswitch has been set to one of the ON positions (**1**), the System Control Module (SCM) must be satisfied that no boards are plugged into incorrect slots, or DC power will not be enabled. Correct board installation is verified by the SCM, which checks each board for correct identification resistor values.

If an incorrectly positioned board is detected, DC power will not be applied to the power buses and an error message is sent to the front panel **SYSTEM STATUS** display indicating which slot has an incorrect board installed.

If all boards are installed correctly, the SCM determines the number of boards installed in the card cage and checks to see if sufficient DC power has been installed. If it finds that not enough power supplies have been installed for the number of boards in the card cage, a hexadecimal error code is displayed on the front panel **SYSTEM STATUS** display and the power-up process stops.

Once correct board installation and DC power supplies are verified, power is applied to both the system power buses and the system held in reset. The SCM then examines the value of each power supply bus. If everything is within tolerance (high or low enough to run the machine, but not so extreme as could cause damage), it takes the PIA and SP2 out of reset, allowing the SP2 to begin EPROM-based POST and the transition to front panel.

3.4.2 Power-On Self-Test (POST)

As the SP2 powers up, it begins a brief EPROM-based Power-On Self-Test (POST) verifying proper operation of the 68000, its local memory, and associated support logic. Upon successful completion of the POST, the SP2 executes *front panel* code, enters front panel mode, and sends the front panel prompt to the console.

During the POST period, the SPU sends reports to the console indicating which subtest is in progress. In case there is a failure in the console interface area of the SPU, the number of the subtest in progress is also reported via four LEDs mounted on the front edge of the SP2 board (visible from the front of the card cage with the card cage cover removed).

If any subtest fails, the failing subtest remains displayed in both places.

3.4.3 Front Panel

Front panel is an EPROM-based monitor—from the moment that the SP2 is reset until the front panel mode is exited, the SP2 is operating out of EPROM.

At this point, the operator can do several different things:

- Run a debugger
- Order extensive SP2 diagnostics

Refer to the *CONVEX Diagnostics Documentation (C200 Series)* for complete details and procedures on the two items above.

- Select alternate boot devices
- Boot only the SP2
- Initiate the automatic boot procedure (SPU UNIX on the SP2, then CONVEX UNIX on the CPU)

Refer to the *CONVEX Processor Operation Guide (C200 Series)* for complete details and procedures on the three items above.

3.5 Fails During Power OFF to Front Panel Transition

A failure in the transition from powerup to front panel can usually be traced to a failure in either the System Control Module (SCM) or the SPU.

If the SCM does not consider the system safe to power up, it will not energize the AC power contactor in the power controller, and the system will remain powered down. The SCM reports any unsafe conditions it detects via a two-digit alphanumeric LED hex **SYSTEM STATUS** display on the system front panel.

If the system fails to powerup, check the **SYSTEM STATUS** display and refer to Chapter 6, "Indicators," for error message explanations and information.

3.5.1 Fails During Powerup

If the SCM is not operating correctly, it may erroneously prevent the system from powering up and not display an error code. It may also apply power to the system but never take the SPU or PIA out of reset, also without the display of an error code on the **SYSTEM STATUS**.

If the PIA is held in reset, there will be no clocks to the SPU and nothing will happen. If the SPU is held in reset, again, nothing will happen. There will be every appearance of a dead SPU—no console activity, no activity on the SPU LEDs.

> **NOTE**
>
> Always check to ensure that the console is connected to the processor and operating properly before assuming that there is a problem with the SPU.
>
> Refer to the *CONVEX Processor Operation Guide (C200 Series)* for complete connection details.

A faulty SPU communication interface on the SCM can also cause problems in the SPU POST and operation.

If SCM problems are suspected, refer to Chapter 6, "Indicators," for information.

3.5.2 Fails During POST

If the SPU POST fails, and there is no indication of trouble on the **SYSTEM STATUS** display (showing *FF*), and assuming no problems with the SCM, then the:

- **PIA may not be sending clocks**—There will be no activity on the SPU LEDs
- **SPU may be faulty**—There is activity on the SPU LEDs, but no activity on the console, or the SPU LEDs indicate a POST number that has failed

3.6 Front Panel to SPU UNIX

Once the system console is showing the front panel prompt, SPU UNIX can be booted from either the SPU disk or the SPU tape drive. The default boot device is normally the SPU disk and booting from the disk is initiated by entering the following command from the system keyboard:

b <return>

The SPU UNIX executive is loaded into SPU memory and SPU UNIX is booted.

During the SPU UNIX boot process, the SPU constantly reports its progress to the system console display. If the process stops at some point, the operator can then see what operation was in progress when the process was terminated.

Next, the SP2 does a file system check (*fsck*) on the SPU disk to ensure file integrity. Once file condition and location has been verified, the files are mounted (access to them is permitted).

At this point, SPU UNIX is up and ready to go, the SPU prompt (`(spu)>`) displayed on system the console.

Once the SPU is booted, the CPU can be booted manually by the operator, diagnostics can be run on the CPU, or any other SPU program can be initiated.

3.7 Fails During SPU UNIX Boot

During the SPU UNIX boot process, the boot loader code directs the SP2 hardware to read the boot track off the SPU disk, store the code in SP2 memory, jump to that location, and begin executing. After a few moments, SPU UNIX is completely loaded and the SPU prompt is sent to the console.

Likely causes for failing during SPU UNIX booting are:

- Corrupted data on the SPU disk boot tracks
- Cabling to the SPU disk
- Faulty SPU disk drive
- Faulty SPU tape drive
- Faulty SCSI interface
- SPU UNIX panic

If there is a failure during SPU UNIX boot, chances are that the cause is probably a SPU disk or SCSI bus problem, or a SPU UNIX problem. Most of the likely SP2 board hardware malfunctions (faulty 68000 processor, SPU memory failures, and so on) should have been detected and reported by the SPU POST.

3.7.1 SP2 Cannot Access Boot Device

If the SPU just cannot seem to get started, there is a possibility that it was unable to access the SPU disk. There could be a cabling problem, the SPU disk drive could be faulty, or the SPU tape drive could be faulty and be locking up the SCSI bus.

Check the cabling.

Then, if a bootable cartridge tape (SPU UNIX boot image) is available, mount it, and then try to boot from the tape:

s b=t <return>

Refer to *CONVEX Processor Operation Guide (C1, C120, C130, C210, C220)* for more information. If the boot from tape is successful, execute *spu2000* and perform a root restore. Next, try to boot from the SPU disk. If the SPU still cannot access the disk drive, it is probably faulty.

If the boot from the tape is unsuccessful, the SCSI bus may be at fault.

Unplug the SPU tape drive from the SCSI cable and try to boot from the disk. If the boot is successful, there is a good chance that a problem in the tape drive is locking up the bus.

> **NOTE**
>
> A bus termination resistor network must be installed on **one** device connected to the SCSI bus for the bus to operate properly. When disconnecting devices to troubleshoot suspected SCSI bus problems, ensure that the remaining device has a terminating resistor network installed.

If the boot from disk is unsuccessful, reconnect the tape drive, disconnect the disk drive, and try to boot from the tape. If the boot is successful, there is a good chance that a problem in the disk drive is locking up the bus.

If the SPU still will not boot, suspect malfunctioning SCSI hardware on the SPU and replace the SP2 board.

3.7.2 SPU Hangs or Crashes During Boot

Booting SPU UNIX will fail if the SP2 is unable to read the boot tracks from the SPU disk correctly. If the boot tracks cannot be read, the boot process stops. If the boot code on the SPU disk is corrupted, if the boot code gets corrupted on the way from the SPU disk to the SP2 memory, or if the SP2 memory is faulty, the SPU will probably hang or crash as it tries to execute invalid boot code. It will likely never get the chance to send a message to the console.

3.7.3 SPU UNIX Panics

If enough SPU UNIX has loaded before an error is detected, the error may cause a SPU UNIX panic message to be displayed on the console as the SPU crashes. Such an error can occur during the later stages of the SPU UNIX process, or during one of the processes that are executed immediately after booting.

If SPU UNIX panics and sends a message, the message should give some indication of the problem. Refer to Chapter 8, "SPU UNIX Messages," for error message definitions and cause and disposition information.

Again, there are only a few things that are likely to be responsible: the SP2 hardware, the SPU disk, or the SCSI bus that connects the two.

3.8 SPU UNIX to CONVEX UNIX

Once SPU UNIX is running (the SPU prompt is displayed on the system console) and no hard error conditions are detected in the CPU, CONVEX UNIX can be booted. If the operator has turned the front panel keyswitch to the **1 SECURE EXECUTION** position, booting of the CPU to CONVEX UNIX is automatic.

If the operator has set the front panel keyswitch to the **1 LOCAL MAINTENANCE** or **1 REMOTE MAINTENANCE** position and issued the automatic CPU boot command (at the front panel prompt), booting to CONVEX UNIX is also automatic.

If, however, the operator has set the front panel keyswitch to the **1 LOCAL MAINTENANCE** position and manually booted only the SPU, the CPU must also be booted manually.

To manually boot the CPU at this point (SPU prompt), enter the following command on the system console:

boot_cpu <return>

Once the command to boot the CPU is received, the SP2 goes through a series of things:

- Checks the clocks and power supplies for nominal performance
- Does a basic reset and initialization of the CPU
- Determines system memory configuration and loads processor register maps accordingly
- Loads processor control stores
- Initializes system memory and check bits
- Loads a portion of CONVEX UNIX device drivers into CCUs
- Loads a small portion of CONVEX UNIX into system memory
- Sets processor program counter to the CONVEX UNIX bootstrap location
- Gates clocks to the system

When the SP2 enables the clocks to the CPU, execution of code in main memory commences, beginning at the pre-set (by the initialization process) program counter location. This starts the whole machine synchronously, which then begins executing UNIX.

3.8.1 System Initialization

The first step in the CONVEX UNIX boot process is the initialization of the CPU.

The CPU must be in a particular state before it can be booted successfully—certain data must be in specific counters and registers before the machine can begin operation. Since the state of the machine at powerup is uncertain, the machine must be initialized to the proper state before operation can begin.

Initialization involves resetting the machine and then scanning (loading) particular bit patterns into certain registers and areas of special RAM in the CPU so that everything is setup correctly before booting is attempted. Depending on which item in the CPU is being initialized, these bit patterns can be determined either by the SPU (based on system resources and configuration), or read from the SPU disk.

The procedures for accomplishing these tasks, as well as most of the data to be used in the tasks, are kept on the SPU disk. Therefore, the pre-boot initialization procedures can be changed to support hardware or system software modifications by simply changing the appropriate software on the SPU disk.

3.8.1.1 Control Stores

Control stores are the microcode or firmware that the VPC and ASP boards use to direct and control their numeric operation. The initialization process loads the appropriate microcode into the special RAM serving each functional area requiring control stores. These RAM systems are not part of the main memory, but are special high-speed RAM systems located near the functions they serve and contain only control store microcode after initialization.

3.8.1.2 System Memory Configuration

The CPU supports a large number of memory board sets, DRAM size and population, and interleave combinations. Memory boards can be populated with 1 Mbit or 256K DRAMs, and memory interleave depends on how many memory board pairs are installed.

Population configuration map registers are located in several functional areas of the machine (SP2, PIA, and CPX). The SPU determines memory configuration and loads these registers accordingly so that the system knows what memory addresses are valid—that these particular memory addresses actually exist in the machine.

Once memory configuration has been determined, the SPU initializes the main memory with good check bits—sets each memory address location to a known data pattern, thereby generating good check bits. This ensures that initial memory check bits are correct and that a memory ECC error is not generated at boot time because the memory is full of random bits.

3.8.1.3 I/O Configuration File

The I/O configuration file contains information on which Channel Control Units (CCUs) are installed in which slot and what type they are (IOP, VIOP, HSP, etc.). This information is maintained by the SPU in a database and is used to initialize the operating system with the current I/O configuration.

Each time a CCU is installed, removed, or changed, the SPU I/O configuration file on the SPU disk must be updated.

Information in this database informs the CPU what kind of controllers are installed and where they are located. This information is passed along to the IOPs so that they will know how to communicate with their devices.

3.8.1.4 CONVEX UNIX *init*

The SPU loads UNIX *init* code into main memory so that when the CPU clocks are enabled, the task of actually booting UNIX and building a UNIX environment can begin.

3.8.1.5 Program Counter

Next, the SPU scans (loads) a particular value into the processor program counter so that it points to the beginning of the *init* code. Then, when the clocks are enabled at boot time, the CPU knows where to begin executing *init* code, and booting can proceed.

3.8.2 Booting the CONVEX UNIX Kernel

Once the machine has been initialized, the CONVEX UNIX *init* code loaded into main memory, and the program counter set to the beginning of the code, the system is ready to boot.

The SPU enables clocks to the CPU which begins executing the *init* code.

3.9 Fails During CONVEX UNIX Boot

There are two parts in the process of booting CONVEX UNIX:

- Initialization
- Execution of the UNIX kernel

The SPU is responsible for the former and the CPU is mostly responsible for the latter.

If the system fails during one of these two processes, the troubleshooting procedure to follow will depend on which part of the process failed. The situation is complicated by the fact that execution of the UNIX kernel may fail as a result of faulty system initialization that was not detected earlier.

3.9.1 Fails During System Initialization

If the CONVEX UNIX boot process fails during initialization, it is relatively easy to determine what is wrong. Since the scan interface is used to initialize the machine, the problem could either be a board in the system or the SPU which is unable to function properly.

When CPU initialization fails, there is a good chance that the problem is in the area that was being initialized when the failure occurs and the message is displayed. This makes it relatively easy to locate the trouble.

For example, if the control store loader breaks and the control stores do not verify, there is a simple one-to-one correspondence between control stores that do not verify and where they are located. Control store loads are scan-based and everything necessary to its success is on that board. Therefore, if there is a mis-verification of a load, the trouble can easily be located.

If the CPU initialization fails, it should be tried again since there are situations in which the machine may not reset the first time a reset is issued after a powerup. Many times, the machine will initialize properly the second time.

If the CPU fails initialization a second time, then it might be a good idea to run the *spu4000* diagnostic test. This test verifies the ability of the SPU to successfully scan the various boards in the system. If this test fails, there will generally be an indication of which board or boards could not be scanned. *spu4000* is discusses below.

There are some initialization processes that are more likely to suffer failures than others. *sysreset* and *mminit* are also discussed below.

3.9.1.1 *spu4000*

spu4000 is a SPU diagnostic test that verifies the operation of the scan subsystem. This diagnostic test checks the scan interface by actually using it on all the different boards. *spu4000* does not attempt to initialize the boards in the system, it only tries to scan them. If it is successful, then the scan interface on the SPU and on the board being scanned are operating correctly.

This test checks the ability of each of the boards to scan in both directions, makes sure that their COP chips are readable, and makes sure that each board that is capable of generating a hard error or soft error can really generate one. If *spu4000* fails, it indicates what it was trying to do when it failed.

Based on results of the *spu4000* tests, which areas of the system are having trouble can be determined and then troubleshooting can proceed accordingly.

If the SPU and its scan interface are found to work for any board, then it can be concluded that it will work for all other areas of the system.

Each functional block in the system (CPU A, CPU B, memory, I/O, etc.) has a separate scan bus to the SPU. If all the boards in a particular functional block are having trouble—are not able to scan properly—then the scan bus to the functional block or the SPU may be faulty.

It is possible that a defective scan interface on a board could be holding the scan bus up or down, thereby causing the initialization (and *spu4000* scan test) to fail on all the other boards in the functional block. If there seems to be a scan bus problem, one way to isolate the board with the bad interface is to start pulling boards in the functional block out one by one until the failing subtest passes.

The last board removed before the subtest passes will most likely have the problem.

To ensure that it is the only board causing trouble, install the other boards and then re-run the subtest. If the subtest still passes with all the other boards installed, then it is safe to conclude that the only board causing trouble has been found. If, however, the subtest fails again, the process of pulling out boards until the subtest passes, re-installing the other boards, and then re-running the subtest should be repeated until no more faulty boards are found.

If *spu4000* does not fail, the trouble is probably on the board that is indicating the initialization problem.

3.9.1.2 *sysreset*

Whenever *sysreset* fails, always try it again before proceeding with other troubleshooting activities—especially if hard errors are reported. The system has been known to generate a hard error on powerup as a result of coming up in some state that will not reset. There have been cases where asserting the **DMODE** line to that board does not clear these errors. When this happens, a lot of messages saying that the SPU cannot scan the system are displayed.

If *sysreset* is executed again, this is usually cleared up. If the message is displayed again, then it is probably valid—the board really will not reset. A board with this problem has suffered a hardware failure of some kind and must be replaced.

sysreset sometimes displays a message indicating that when it tried to force the ASP into a microcode loop to go out and initialize the caches, the operation failed. The ASP is most likely at fault when this happens.

There are specific error messages that will be displayed by *sysreset* if a particular board did not reset correctly. It is probably a good indication that there is something wrong with that board.

3.9.1.3 *mminit*

mminit tries to verify that installed system memory is configured (Memory Array Module (MAM) and interleave) correctly. If it is unable to verify a valid memory configuration, it issues warning messages. If it is able to establish *any* valid memory configuration, it will configure the system and display a message accordingly.

This may cause a portion of the system memory to be unavailable. For example, if *mminit* determines that a Memory Control Module (MCM)
board has the wrong size MAM, it will configure for the largest valid size that it can use. This condition may be caused by a MAM of one size that looks smaller to *mminit* because it is defective. In this situation, *mminit* configures the memory half as large as it really is, proceeds with the initialization, and reports its activities.

If *mminit* fails, run it again with the *-s* option. In this mode, *mminit* directs the SPU to perform all the tasks directly, and does not use any CPU facilities. If *mminit* fails, but is successful with the *-s* option, a CPU problem is indicated.

3.9.2 Fails During UNIX Kernel Execution

If the CONVEX UNIX boot process fails during UNIX kernel execution, it is somewhat more difficult to determine what is wrong. Since the successful booting of the UNIX kernel depends on the earlier successful system initialization by the SPU, the problem could either be in board in the CPU in the SPU which was unable to initialize the machine properly.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4
Fails During UNIX Process Execution

4.1 Overview

This chapter explains the activities involved in running a UNIX process in general, and discusses what might go wrong, causing the machine to fail and display a UNIX panic message. Guidelines on troubleshooting a machine that has failed while running a SPU UNIX or CONVEX UNIX process are provided.

4.2 UNIX Processes

This section provides a general overview of how UNIX operates on a system.

TBD

4.3 Fails During SPU UNIX Process

This section details how SPU UNIX might fail while executing a process and provides information on how to troubleshoot a system that has failed.

TBD

4.4 Fails During CONVEX UNIX Process

This section details how CONVEX UNIX might fail while executing a process and provides information on how to troubleshoot a system that has failed.

The possible failures while executing a process under CONVEX UNIX have been grouped into five categories:

- **Hang**—System missed a transaction and waits indefinitely
- **UNIX Crash**—UNIX kernel is lost
- **Hard Error Crash**—Invalid data is detected
- **Wrong Answer**—Invalid code, design problem
- **Wrong Answer/Core Dump**—Illegal operation

Each of these is discussed below.

4.4.1 Hang

A system hang occurs when a machine function misses a transaction. Since the machine must wait for the transaction to complete before processing the next instruction, all processing stops while it waits for a transaction that has already completed.

4.4.2 UNIX Crash

A UNIX crash occurs when the UNIX kernel is lost.

4.5 Hard Error Crash

A hard error crash occurs when the machine detects invalid data (parity errors) in a register or in data moving across a data bus.

4.6 Wrong Answer

A wrong answer error can occur as a result of an Operating System (OS)
bug, an application program bug, invalid microcode, or microcode and board set revision incompatibilities.

4.7 Wrong Answer/Core Dump

A wrong answer and core dump usually result from an illegal operation that has been requested by the OS or application program (divide by 0, for example).

Chapter 5
Fails Diagnostics

5.1 Overview

This chapter discusses the different ways that a machine can fail a diagnostic test and what elements might be involved in the failure. Explanations of the different machine failure modes that lead to diagnostic testing and guidelines on using and interpreting diagnostics to troubleshoot machines with those failures are provided. This includes machines that get wrong answers, machines that suffer a failure while running an application program but that do not fail diagnostic tests, and machines that repeatedly fail diagnostic tests.

The interaction of the various diagnostic subtests with the machine hardware and the effect that the machine's distributed architecture has on the effectiveness of the diagnostic tests is also discussed.

The user is directed to the *CONVEX Diagnostic Documentation (C200 Series)* for additional information concerning diagnostic testing and interpretation of messages generated by the diagnostic tests. Appendix A, "CPU Instruction Glossary," of this manual lists the CPU instruction abbreviations that might appear in a diagnostic message along with their definitions.

5.2 Troubleshooting With Diagnostic Tests Flowchart

The following flowchart details the typical sequence of events for isolating failed boards using diagnostic tests:

TBD

5.3 CONVEX C200 Diagnostic Tests

> **NOTE**
>
> For a more rigorous discussion of CONVEX C200 diagnostics, refer to *CONVEX Diagnostic Documentation (C200 Series).*

CONVEX C200 diagnostic tests (diagnostics) are suites of functional test programs which, in most cases, execute under SPU UNIX. These programs use the facilities of the SP2 to access and test the operation of various areas of the machine and report any errors that are detected.

There are five test categories:

- **CPU**—CPU subsystem
- **DEV**—Peripheral device
- **I/O**—I/O Subsystem
- **MEM**—Main memory subsystem
- **SPU**—Service processor subsystem

Within each of these categories, there are various test programs that test and verify major areas of the subsystem under test. And within each of these tests, there are a number of subtests that verify the operation of a specific machine, peripheral, or SP2 function. A number of related subtests are sometimes grouped into a *class.*

The subtests are arranged within the test such that the most basic functions are tested and verified first. Then, the more complex operations (which are made up of the basic functions verified earlier) are tested. Finally, the most complex functions are tested and verified.

5.3.1 CPU Diagnostic Tests

CPU diagnostic tests should be run in the correct order (from the least complex to the most complex) to get the best use from them. As the test progresses, each subtest assumes that the functions tested by previous subtests are working properly. So when the vector add function is tested, for example, the ability to load the vector registers correctly has already been tested and verified.

When a subtest fails, it is not always conclusive that the function being tested is broken. It could be anything in the chain leading up to that subtest that did not work correctly. And if the tests are executed in the proper order from the beginning, there is a better chance of being able to locate the failing item quickly.

If the most complex test is run first and it fails, there are a vast number of possibilities.

Failure of any subtest is reported. As a general rule, the message displayed when a test fails does not indicate that a particular Field Replaceable Unit (FRU) should be replaced. Instead, the subtest that failed along with the instruction or operation that was tested is indicated. At the conclusion of diagnostic testing, analysis of which basic functions failed provides insight into where the failed hardware is located so that repair can be effected.

Because improvements are constantly being incorporated into both the system hardware and the microcode, there will be rare times when a subtest fails when nothing is wrong. When this situation is discovered, it is documented in the release notes until the hardware or microcode are upgraded.

As a general rule, however, when a subtest fails, it can be safely concluded that there is a problem with the system.

5.3.1.1 Isolation of Functional Blocks

From the diagnostic testing point of view, there is less than optimum isolation between CPU boards in the processor and between functional blocks within the processor. This makes identifying the correct board to replace when subtests fail more difficult.

When a CPU diagnostic is testing a function and it fails, the trouble could be in one or more functional blocks and on one *or more* boards of the CPU. In a machine with multiple processors, the trouble could be in any (or several) of the processors.

If a particular subtest fails, for example, it does not necessarily mean that there is anything wrong with the instruction in the processor that is being tested. It could be that trouble in another processor caused it to write invalid data into a memory location, or to write valid data into the wrong memory location, causing the tested instruction to fail. Diagnostic tests are not able to detect this type of situation.

It is also possible that trouble in one function could cause another function to fail its test.

5.3.1.2 Running Diagnostic Tests

Diagnostic test suites are structured such that they build up a history. As each subtest passes, subsequent subtests assume that everything up to that point has been successful.

However, this may not always be the case—the previous subtest may not completely test the previous function. When a previous test did not detect a fault in a function, a subtest some time later may fail. This is not because the instruction it was testing has failed, but because imperfect testing of a previous function did not detect a fault in a function needed for the current instruction to execute properly.

The diagnostic tests require valid control stores to have been loaded and valid data to be in the memory before the test is run. The information that is loaded into the vector registers for an add test, or whatever, is put into memory by the diagnostic during an earlier test.

If there is strong evidence that a particular function of the machine is faulty, it might be a good idea to run the diagnostic subtest or subtests for that particular function. However, if tests fail, it is less conclusive than if all the subtests up to that point had been run and passed before the current subtest failed.

5.3.1.3 Interpretation of Diagnostic Test Failures

When a diagnostic test fails, what really happened? The actual error or failure can be displaced some distance in both time and space from the instruction that fails during a diagnostic test.

Knowledge of the system architecture, instruction set, and the diagnostic tests are the best tools to apply to the task of ambiguous test results.

Processor conditions can cause misleading diagnostic test results the same way that they can cause misleading error messages to be generated by the operating system while executing a process. Something could have failed a long time ago and written invalid data somewhere that is not used until later by a completely different function in a different part of the processor—or in a different processor. The UNIX error messages (and diagnostic subtest results) indicate that an error has been detected, but there is no conclusive indication when the error actually occurred or what function caused it—only what function was in progress when the error was *detected.*

The following scenarios illustrate situations in which the function failing the subtest is not the function that is broken:

- A faulty memory system can change an instruction in storage that might not be retrieved until a particular subtest which used the instruction is executed. When the subtest fails, it would not be the function being tested that is broken, but the part of the memory system responsible for changing the *add* instruction to a *nop* or *jmp*. As a result of the memory system failure, the diagnostic subtest did not execute properly.
- The SPU loads data and code from the SPU disk into the processor and memory systems before diagnostic tests can be run. It is possible that this data can arrive at its destination corrupted. This can be caused by data being incorrectly read from the SPU disk, written into the system incorrectly, or, possibly, the data on the SPU disk is corrupted. Then when a diagnostic subtest is executed, it fails as a result of the invalid data.
- Before the vector floating point add test, data is loaded into vector registers. Then the add is executed and the result examined. If one of the loads did not work properly, the add operation fails. In addition, just about any other subtest requiring this function to be operational will also fail.

In each case, it would be difficult to determine if the instruction that was being tested failed because it was faulty or if it failed because it had bad data to work with or because some other instruction failed.

It is at this point when whatever knowledge of the system, how it works, what is going on when instructions are executed, where instructions are executed, where data comes from, and where it goes will be the most benefit to the Field Engineer (FE)—or whomever is attempting to troubleshoot and repair the system.

5.3.2 Machine Malfunctions

There are two general categories of machine malfunctions:

- System failures
- Wrong answers

5.3.2.1 System Failures

A system failure is a machine malfunction that results in a system hang or a system crash. A system failure may occur:

- During initial power up
- At front panel
- During SPU UNIX boot
- During execution of a process under SPU UNIX
- During CONVEX UNIX boot
- During execution of a process under CONVEX UNIX

For a more detailed discussion of these activities, refer to Chapter 3, "Fails During Boot."

There will always be some indication that a malfunction has occurred after a system failure, and there will usually be some evidence to examine (error logs, etc.) which will be helpful in determining the best troubleshooting course to pursue.

As a general rule, after ensuring system revision level compatibilities, diagnostic tests are run to try to isolate the failure to a Field Replaceable Unit (FRU) which can then be replaced.

5.3.2.2 Wrong Answer

If a customer's machine appears to be operating normally, but the application program being executed gets the wrong answer, the error is much more difficult to successfully troubleshoot than a system failure. In addition, the program may get the error on some runs with a particular data set and not on others with the same data set.

The usual troubleshooting tools may not be of any use since diagnostic tests may not fail in this situation. In addition, the customer may not have realized that there has been an error until many hours after the program run completed. And by then it is too late to determine the state of the machine at the time of the error.

Sometimes the cause of the error turns out to be a bug in the operating system or the application program. And sometimes it turns out to be a design problem in either the hardware or the microcode. The result of the former is up to the software manufacturer. The result of the latter is new microcode or a hardware Engineering Change Notice (ECN) going into the machine.

If an application running on the system gets a wrong answer, the standard troubleshooting procedures (diagnostic tests, board swapping, etc.) may uncover a failure that can be fixed by repairing the machine.

The machine, however, may pass all diagnostic tests. In this case, a copy of the application program getting the wrong answer and the data set in use must be returned to CONVEX for analysis. Sometimes the error cannot be duplicated at the factory.

Consult the Technical Assistance Center (TAC) for details and instructions.

5.4 Troubleshooting With Diagnostics

Once it has been determined that the system suffered a failure and an area of the machine has been identified as the possible cause of the problem, appropriate diagnostic tests are selected and executed to reveal any hardware failures.

5.4.1 Diagnostic Tests Do Not Fail

When a system has failed, but all diagnostic tests pass, there are usually two explanations:

1. The failure is intermittent and did not happen to occur during the execution of diagnostic tests.
2. The diagnostic tests cannot duplicate the machine conditions causing the failure.

Neither situation permits the normal troubleshooting procedure to be pursued.

Some educated guesses can be made as to what was going inside the machine when the failure occurred and boards swapped accordingly. However, this is a tedious process and may not clear up the problem. This may be the result of a design problem—the boards may be functioning exactly as designed.

There is probably no choice in these situations but to send a copy of the application program and the data set causing the failure to CONVEX for analysis. Even so, the failure sometimes cannot be duplicated at the factory.

Consult the Technical Assistance Center (TAC) for details and instructions.

> **NOTE**
>
> It is possible that a borderline function that does not normally fail diagnostic tests can be made to fail if the system clocks and DC power supplies are margined and the tests run again.

5.4.1.1 Intermittent System Failures

When the system fails during the execution of a particular application and data set combination some times, and other times does not, there is probably an intermittent hardware problem.

This situation is particularly difficult to troubleshoot successfully.

Again, some educated guesses can be made as to what was going inside the machine when the failure occurred and boards swapped accordingly. However, since the failure is intermittent, this is a tedious process and one can never be sure that a board swap has cleared up the problem until the failure *does not* show up again.

The approved procedure, is to take a crash dump after the system has failed and send it CONVEX for analysis and then (if practical) run diagnostic tests as often and as long as possible until the failure occurs again.

5.4.1.2 Repeatable System Failure; All Diagnostic Tests Pass

Sometimes the system fails every time a particular application program and data set combination is executed, but passes all diagnostic tests.

This can be caused by some obscure and unique operating system, application program, and data set processor activity (unusual sets of timings and events) that is simply not duplicated by the diagnostic tests. There are so many possible permutations of instruction combinations and timings that it is not possible to duplicate all of them in diagnostic tests, much less run the tests in any reasonable length of time if they could be duplicated.

5.4.2 Diagnostic Tests Fail Intermittently

Transient hardware problems can be responsible for intermittent failure of diagnostic tests—a subtest will fail once every 10 or 15 times it is run. Diagnostic tests may not be able to isolate these problems.

In this situation, when a diagnostic test finally fails, it will give some indication of what instruction was being executed when the test failed. *The failing instruction is not necessarily the cause of the intermittent system failure, however.* It may just be what was being tested when the intermittent failure occurred in another part of the system and caused the instruction to fail.

Then, because of the transient nature of the initial hardware problem, it will be difficult to verify the repair after boards are replaced.

5.4.3 Diagnostic Tests Fail Repeatably

When a diagnostic test fails, there is at least some indication of where to start looking for the problem. A failed diagnostic subtest generally means that the specific function that it was trying to execute failed to work as expected. This can be caused by one or more reasons, which will generally fall into the following categories of failing diagnostics:

- Revision incompatibilities in a microcode and board set
- Design problem
- Failed hardware

When a test fails, there is the possibility that the microcode resident in the system when the test was run had somehow been corrupted without generating a hard error. To help avoid needless troubleshooting of nonexistent failures, always re-load control stores and re-run the test before proceeding with additional troubleshooting.

If evidence indicates a microcode problem, the first thing to do is to try to re-load the microcode. Execute the control store loader by entering the following command at the `(spu)>` prompt:

```
cs
```

The loader takes about 2 minutes per processor and returns the SPU prompt when complete.

After the microcode is reloaded, run the failing diagnostic test again.

If the test still fails, the problem is probably in one of the other areas (incorrect revision level of microcode, incorrect revision level of boards, failed board, or design problems).

5.4.3.1 Revision Level Incompatibility

The revision level of the installed microcode must be compatible with the revision level of the installed board set. If there are incompatibilities between the microcode and the board set, there could be either solid or intermittent problems (system crashes, incorrect answers, failed diagnostic tests, etc.).

There can be problems, for example, if some boards in a CPU were changed as as result of repairs to the machine or an uprev procedure and the microcode was not upgraded appropriately. Likewise, if new microcode is installed on an old board set, there could be problems that may not turn up until later. There have been several problems in the field that have been fixed by simply upreving the microcode or boards so that the combination was compatible.

Check the system configurator to make sure that the microcode and the board set that are installed in that machine are compatible. At this time, this must be done manually, as described below. However, a program containing configurator information is being planned for the SPU which will check and verify compatibility.

If the revision levels of the installed microcode and the installed board set are shown to be compatible on the hardware configurator, then it is likely that the system problems are not due to revision level incompatibilities. There is a slight chance, however, that the system under test has an unknown incompatibility.

System Configurator

The system configurator is a document issued by the hardware group detailing the combinations of microcode and hardware that are compatible.

There are a number of configurators (I/O, CPU, etc.) for each of the different CONVEX C200 Series machines—the appropriate one must be consulted (refer to *Tech Tips*).

Microcode Revision Level

There is no generic utility that reports the revision level of the installed microcode. However, the revision level can be determined by checking the *mnt/usr/UCODE* file on the SPU disk. This file is updated when new microcode is installed from the SPU tape.

Board Set Revision Level

Each board in the card cage is equipped with an Electrically-Erasable Programmable Read-Only Memory (EEPROM) in which is stored various identification and revision information about the board. This EEPROM is commonly called the COP chip, and is accessible to the SPU via the scan bus. Since COP chips are read by the SPU through the use of the scan interface, the COP chips cannot be read while the machine is running (a crash would result).

When SPU UNIX is booted, it directs the SPU to read the COP chip on each board in the system, and place the result in the **/mnt/usr/scan/cop.out** file on the SPU disk. The COP information is then accessible at any time.

5.4.3.2 Design Problem

If there are no known revision level incompatibilities found, that leaves design problems and failed boards. The next thing to check for is known design problems. It is possible that otherwise compatible microcode and board set installed in the system under test could have failures that have been uncovered since the configurator was prepared.

Report the microcode and board set revision level to TAC and inquire if there are any known problems with the combination.

There have been a few system failures in the field that were traced to hardware design or microcode problems. These problems have been very obscure and tended to surface only in some of the affected parts—usually as a result of unusual processor activity some time after initial fielding. Known problems have been carefully tracked and corrected in later revs of the affected boards and microcode.

5.4.3.3 Failed Hardware

If there do not seem to be a revision level incompatibility or any known design problems in the system under test, it may be safe to conclude that the system has suffered a hardware failure of some kind in the function that was being tested when the diagnostic failed.

If a memory diagnostic test fails, chances are that it is either the memory board indicated in the error message or the SPU. At this point, it is usually a matter of running the SPU tests and then changing whichever board is indicated and running the test again.

If a PIA diagnostic test (I/O, DEV), CPX, or SPU diagnostic test fails, it is relatively conclusive.

If a CPU test fails, however, it is more difficult to locate the faulty board since there are at least six possibilities in the CPU (ASP, DCU, IPP, SFU, VPC, and VPD) as well as problems on other boards (memory, SPU, etc.) that could affect the CPU tests. As a general rule, it is going to take knowledge of the machine and how it operates, as well as knowledge of the diagnostic tests and what they do when they run, to isolate the problem down to a board that can be replaced.

If just one subtest fails, and all others pass, it is relatively easy to identify physical location of the problem since there is considerable overlap of the instructions that are being used over and over again.

As a general rule, however, a number of diagnostic subtests will fail (as opposed to just one subtest failing) as a result of any given CPU hardware failure. A number of failed subtests, along with the knowledge of what the subtests were doing when they failed and knowledge of the machine and what it was doing while performing the tests will help triangulate on the specific failed area.

If a large number of the subtests fail, however, it may be difficult to determine what they have in common and what is different. Depending on how many spare boards are available, it might be faster to swap boards and re-run the diagnostics that failed than to try to figure out all the possible permutations and combinations of failing and passing subtests to isolate the problem to a board.

If some of the simpler subtests fail (the ones that are run first)—instructions cannot be dispatched, for example—it is very difficult to determine the physical location of the problem since so many functional blocks are involved. The further the diagnostic test suite progresses before there are failures, the easier it will be to determine which board to replace when a subtest finally does fail due to the increasingly narrow field of view as the test suite progresses.

Then, again, sometimes it is easy to determine (with a high degree of confidence) if it is a scalar or vector problem. If it seems to be a vector problem, then only two boards (VPC and VPD) are likely suspects. If it seems to be a scalar problem, however, then it is likely one or more of the scalar boards (ASP, DCU, IPP, SFU), or possibly the PIA or CPX.

5.4.4 Repair Verification

After failing diagnostic tests and other information have been used to isolate the hardware failure down to a board (or boards) and the suspect board(s) replaced, the previously failing diagnostic tests should be run again to verify the repair.

If all the previously failing tests now pass, a series of top-level CPU diagnostic tests should be run to ensure that all problems have been found and fixed. For the CONVEX C210:

- *cpu4030*
- *cpu4231*
- *cpu4233*
- *cpu4041*

> **NOTE**
>
> Consult *CONVEX Diagnostic Documentation (C200 Series)* for information about these tests and to determine the equivalent tests and their appropriate modes for the system under test.

These tests verify the basic scalar and vector instructions and also run multiple processors if there are multiple processors installed. When run in the correct mode (parallel, chained, and for *cpu4041*: vl = 16), these tests should take about 10 minutes to complete, and provide a high degree of confidence that the problem has been fixed.

If a system failure during the execution of an application program was the cause of the service call, that application should now be run exactly as it was when it failed (same operating parameters and data set). If the system does not fail, it may be concluded that the problem has been fixed.

Chapter 6
Indicators

6.1 Overview

This chapter details the various system indicators, explains the operation of the System Control Module (SCM) and its purpose in the system, discusses the power controller and its indicators, and lists each front panel **SYSTEM STATUS** hex display error code with a definition and a cause and disposition.

6.2 Introduction

The system displays many aspects of its status via front panel indicators.

There are three LED indicators and a two-character alphanumeric LED display on the processor cabinet front panel, and five LED indicators on the processor cabinet door panel. The System Control Module (SCM) drives these indicators in response to conditions it finds during its system checking and monitoring activities.

The power controller indicates AC power input to the processor cabinet and AC power to the DC power supplies via indicator lights on its front panel.

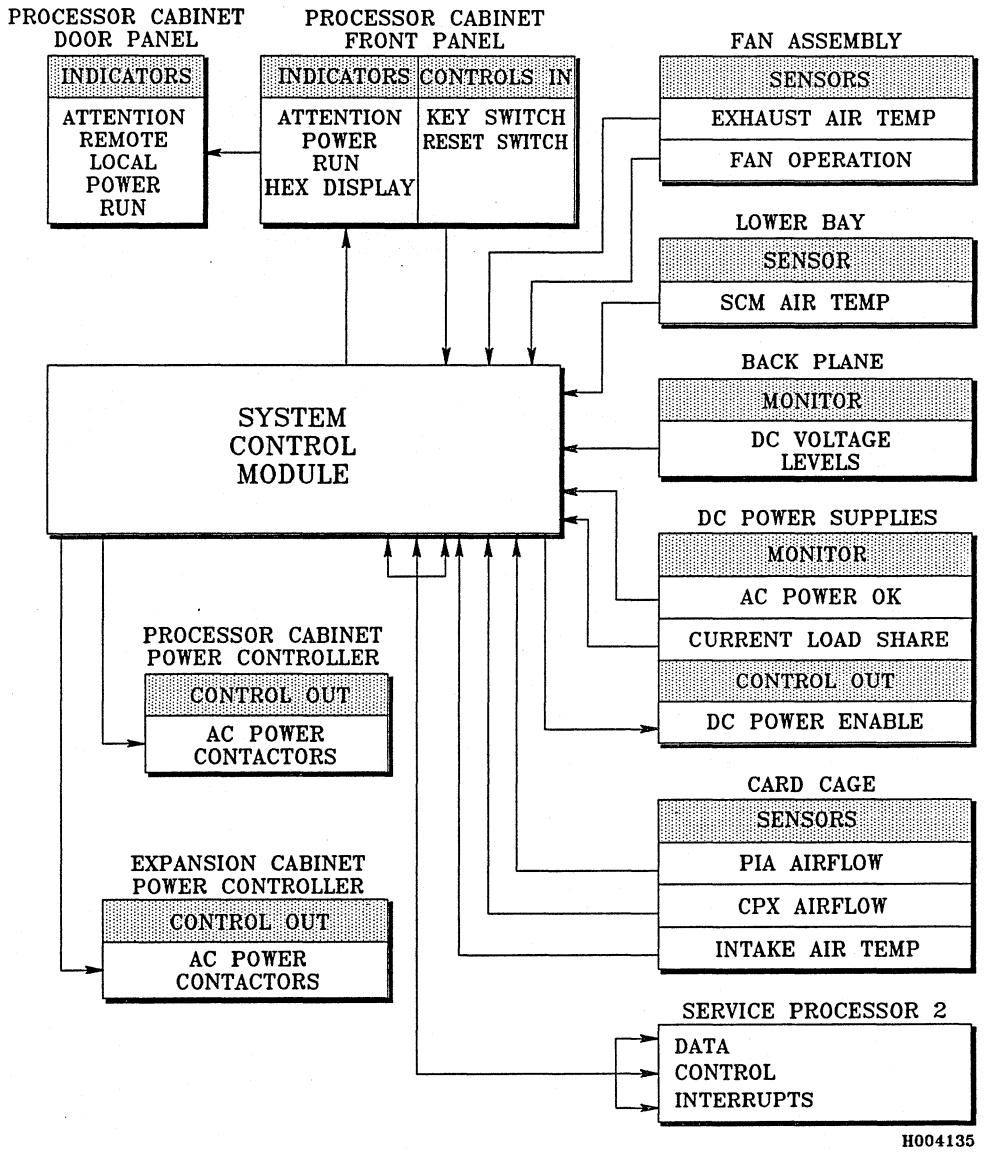
6.3 System Control Module (SCM)

The System Control Module (SCM) determines the system configuration before power up and then monitors AC power, DC power, system temperature, and internal airflow while it is in operation. The system status is determined as a result of this measurement activity and is reported via the front panel and door LED indicators and the front panel hex display.

The SCM also interprets the front panel switches and controls the application of AC and DC power to the system. The system status is the basis for allowing the system to be powered up, and for safety shutdown of the system.

The following block diagram illustrates the SCM's areas of influence and responsibility:

Figure 6–1, System Control Module



6.3.1 Pre-Power Up

The SCM assesses several system parameters before allowing the machine to power up. The SCM, however, is powered up whenever the processor cabinet is connected to a source of AC power and the main circuit breakers on the power controller are set to **ON**.

When first powered up, the SCM initializes all internal writable registers, clears all internal state machines, and clears its onboard *deadman* timer. The deadman timer shuts the system down in the event of SCM failure. It is a 4-bit counter that is clocked from the 60 Hz line and is reset periodically as a result of normal operation of the onboard microprocessor. If the reset from the microprocessor does not arrive in time, the timer rolls over. The carry output from the timer shuts the system down if it is running or prevents power up if it is not, and causes *00* to appear on the front panel hex status display.

Next, the SCM determines the system configuration.

The SCM scans each slot in the card cage, and the identification (ID) resistor network of each card found to be installed is measured to ensure that no boards are installed in incorrect slots. If any incorrectly installed boards are found, the SCM displays messages on the front panel hex status display indicating which slots have incorrect boards installed. System power up is inhibited until all incorrect board readings are cleared.

As each installed board is identified, the SCM keeps a running total of the power requirements for each board correctly installed in the system. When board polling is complete, the SCM checks the 'XIST lines from the power supplies to determine what DC power supplies are installed in the system.

The SCM then compares the power requirements of the installed board set with the number of DC power supplies it found to be installed to determine whether or not there is sufficient DC power to run the board set. If it finds DC power to be insufficient, system power up is inhibited and the *C0* message is sent to the front panel hex display.

Once the initial system configuration is complete, the SCM continues to monitor the machine state to detect any changes that might be made after initial SCM power up. Monitoring of the front panel keyswitch begins at this time.

6.3.2 Power Up

When a power up command is received from the front panel keyswitch (keyswitch is set to any of the three **ON** positions), the SCM once again verifies the board set and DC power supply configuration. If all is found to be in order, the SCM energizes the power contactors in the power controller to apply AC input power to the DC power supplies. After waiting 2 seconds for things to stabilize, the SCM checks the ACPOK line from each of the supplies.

If any power supply reports incorrect AC power or is unable to demonstrate control of its ACPOK line, the SCM de-energizes the power contactors, exits the power-up routine, and sends a message indicating the power supply reporting bad AC power to the front panel hex display. The SCM will not allow another attempt at powering up the machine until the front panel keyswitch has been cycled to the **OFF** position.

If no power supply reports bad AC power, the SCM enables DC output from the supplies, one current sharing group at a time, with about 250 ms between groups. About 250 ms after the last power supply has been enabled, the SCM closes its onboard peripheral cabinet power relay, asserts the reset lines to the SP2 and the PIA, and then waits another 250 ms before beginning to monitor the operating state of the machine.

After all power supplies are on and have stabilized, the SCM checks each one for correct AC power, output voltage levels, and current sharing levels. If any are found to be out of specification, the system is shut down and the appropriate message sent to the front panel **SYSTEM STATUS** hex display.

> **NOTE**
>
> The +12V, -12V, and -5V power supplies cannot be remotely enabled—they are powered up as the power controller AC contactor closes.

If the SCM finds that all power parameters are within tolerance, it de-asserts the reset line to the PIA, allowing it to send clocks to the SP2. Then, 100 ms later, it de-asserts the reset line to the SP2.

At this point, the SCM begins normal system monitoring.

6.3.3 Normal Monitoring

The SCM conducts normal monitoring as long as the system is powered up. The purpose of normal monitoring is to detect potentially hazardous system conditions, and then warn the operator that the problem exists or shut the system down (if the situation is serious) to prevent damage.

Normal monitoring consists of evaluating the following items:

- AC power into the DC power supplies
- DC voltage levels
- DC power supply current load sharing levels
- Input, exhaust, and lower bay air temperatures
- Internal airflow
- Fan operation

6.3.3.1 AC Power

The SCM scans the ACPOK line from each DC power supply in response to a power system interrupt. If one is indicating bad AC power, the machine is shut down and a message is sent to the front panel hex display indicating the power supply reporting the problem.

6.3.3.2 DC Voltage Levels

DC voltage levels are monitored at the backplane. Out of tolerance levels fall into three categories:

- A voltage level greater than ±3%, but less than ±7% of nominal causes a warning message in the format *A#* (where # indicates the voltage level which is out of tolerance) to be displayed on the front panel hex display. The system remains up as long as the ±7% value is not exceeded.
- A voltage level greater than ±7% of nominal causes system shutdown and an error message in the format *A#* (where # indicates the voltage level which is out of tolerance) to be displayed on the front panel hex display.

> **NOTE**
>
> Warning level is extended to ±7% and shutdown level is extended to ±10% when power supplies are margined; however, correct operation of the machine is not expected at those levels.

- A voltage level of 0 VDC to -10% of nominal causes immediate system shutdown and a message in the format *8#* (where # indicates the voltage level suffering the failure) to be displayed on the front panel hex display. Hardware comparator circuits on the SCM monitor this parameter and send an interrupt directly to the SP2 when tripped.

6.3.3.3 Current Load Sharing

All -4.5V power supply outputs are strapped together to provide adequate current capacity for the machine. The same is true for the -2V power supplies (if two are installed). Each of these power supplies reports its current load to the SCM, which scans these values as part of normal monitoring.

The SCM compares the current load of all power supplies of a particular voltage level (-2V or -4.5V) to ensure that they are all within ±10% of each other. The power supply with the lowest load is identified, and that value is considered to be the lowest acceptable current load. Next, 25% is added to that value to establish the upper value for current load. Then the SCM checks to see that the current load for all power supplies in that voltage level fall into the window.

If a supply reports a load above the high limit of the window, it is considered out of tolerance and the SCM shuts the system down and sends a code to the front panel hex display indicating the power supply reporting the high load.

If more than one supply reports a current load above the high limit of the window, the SCM concludes that the lowest value is out of tolerance, and shuts the system down and sends a code to the front panel hex display indicating the power supply reporting the lowest load.

6.3.3.4 Air Temperatures

Three thermisters measure system air temperature. One is mounted on the bottom of the card cage and senses the temperature of the intake air. Another is mounted on the fan assembly and senses exhaust air temperature. The third is mounted on the SCM board and measures lower bay air temperature.

The SCM scans the thermisters and compares the temperatures that they report to warning and shutdown values. If it finds one in the warning range, the SCM sends a *WARNING* message to the front panel hex display indicating which thermister reports over temperature. The system remains up.

If any thermister reports air temperature over the shutdown threshold, the SCM shuts down the machine to avoid over temperature damage and sends a message to the front panel hex display indicating which thermister reports the shutdown temperature condition.

Warning and shutdown temperatures for each location are shown in the following table:

Table 6–1, SCM Temperature Trip Points

Location	Warning	Shut Down
Intake	100 °F (38 °C)	113 °F (45 °C)
Exhaust	131 °F (55 °C)	144 °F (62 °C)
SCM	131 °F (55 °C)	144 °F (62 °C)

6.3.3.5 Internal Airflow

There are two internal airflow sensors: one on the CPX board and the other on the PIA board. These sensors monitor the airflow through the processor card cage.

The SCM scans these sensors (if their boards are installed), and if one reports insufficient airflow, the SCM shuts down the machine to prevent over temperature damage and sends a message indicating the sensor reporting the insufficient airflow condition to the front panel hex display.

6.3.3.6 Fan Operation

A fan sensor (either an airflow or a fan rotation sensor, depending on revision level) is mounted on the fan assembly near each of the six fans. These sensors monitor the operation of the fans.

The SCM scans these sensors, and if *ONE* reports a fan failure, the SCM sends a *WARNING* message to the front panel hex display indicating the failed fan.

If *MORE* than one fan sensor reports a fan failure, the SCM shuts down the machine to prevent over temperature damage and sends messages to the front panel hex display indicating which sensors are reporting failed fans.

6.4 Front Panel and Door Panel Indicators

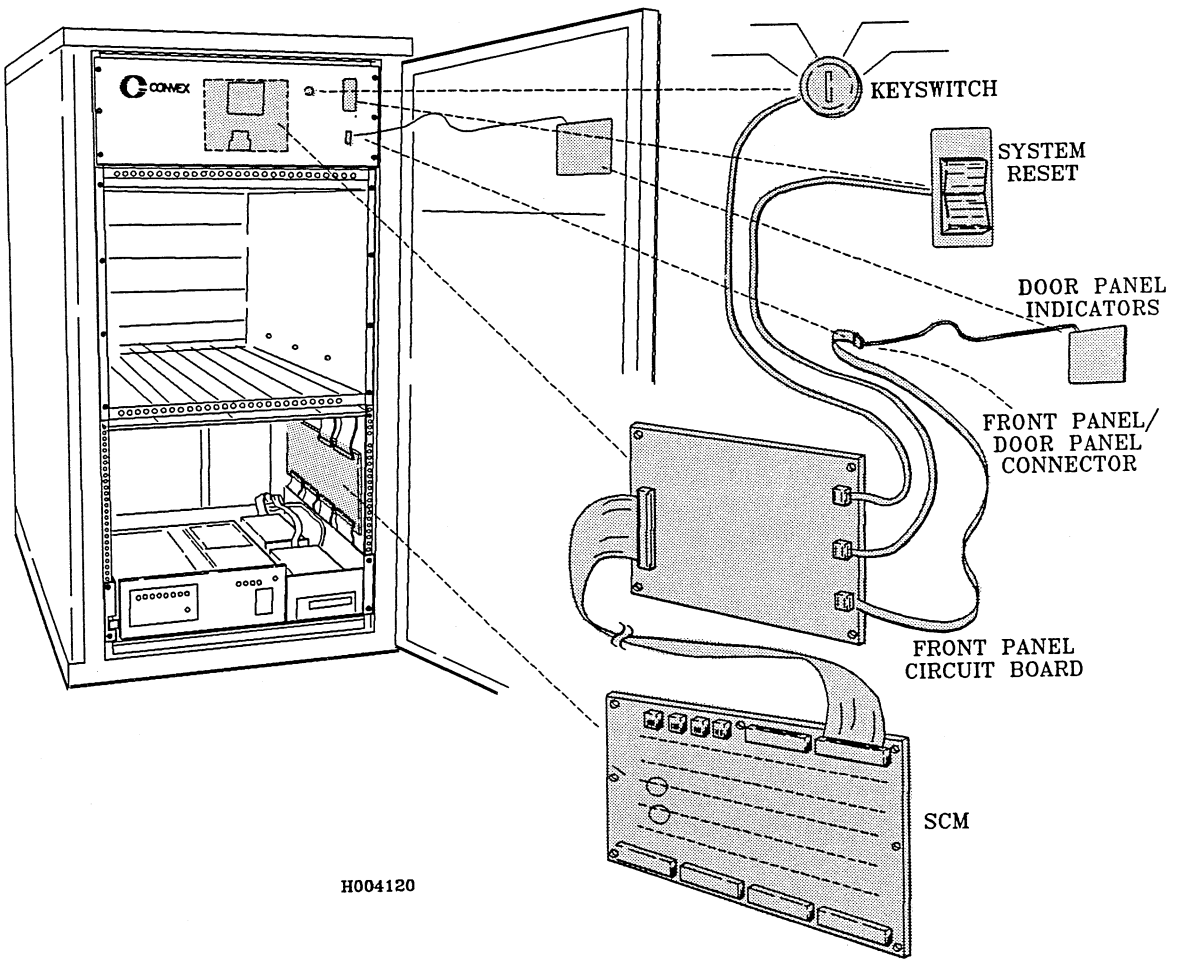
The SCM drives the three front panel indicator LEDs and the five door panel indicator LEDs.

The front panel is connected to the SCM via a ribbon cable. This cable plugs into the front panel circuit board on which are mounted the front panel indicators and the hex status displays, and to which are connected the front panel keyswitch and the front panel system reset switch.

Another cable runs from the circuit board to a connector receptacle on the front panel. A cable from the door panel indicators plugs into this connector.

The following figure illustrates the cabling associated with the front panel indicators:

Figure 6–2, SCM-to-Indicator Cabling



The following indicators are on the front panel:

- **ATTENTION**
- **POWER**
- **RUN**

The following indicators are on the door panel:

- **ATTENTION**
- **REMOTE**
- **LOCAL**
- **POWER**
- **RUN**

The indicators common to both the front panel and the door panel are driven by separate bits of the same register on the SCM to ensure sufficient drive power. SCM firmware controls both bits of the register simultaneously, so the two indicators should always be in agreement.

If the status of the door panel indicators do not coincide with the front panel indicators, check for wiring problems between the front panel circuit board and the door indicators. There could also be a hardware problem on the SCM or the LED could be faulty.

6.4.1 ATTENTION

The **ATTENTION** indicator is controlled by the SCM and will flash in response to the status of several system areas:

- The machine has been margined
- A voltage out of tolerance warning (A#) is being displayed on the front panel hex display
- The SP-SM.ERROR line from the SP2 has been asserted

Malfunction of this indicator could be caused by a problem in the SCM or SP2, the wiring to the front panel or door panel, or by failure of an LED.

6.4.2 POWER

The **POWER** indicator is illuminated when the SCM has energized the power controller AC contactors. Malfunction of this indicator could be caused by a problem in the SCM, the wiring to the front panel or door panel, or by failure of an LED.

6.4.3 RUN

The **RUN** indicator reflects the state of the SP-SM.RUNLED line from the SP2. The bit in the SP2 register controlling this line is set by the SPU UNIX *errintd* process to indicate that CONVEX UNIX is running on the CPU. If the system fails, *errintd* resets this bit, extinguishing the indicator.

Malfunction of this indicator could be caused by a problem in the SP2, the wiring from the SCM to the front panel or door panel, or by failure of an LED.

6.4.4 REMOTE

The **REMOTE** door panel indicator reflects the status of the front panel keyswitch (**1 REMOTE MAINTENANCE**). However, it is not driven directly by the keyswitch. Rather, it is driven by a SCM register that has been written with the results of reading the status of the keyswitch.

When the keyswitch is set to **1 REMOTE MAINTENANCE**, the system accepts commands only from the remote console connector. A modem is usually attached to this connector, allowing the system to be operated from an offsite location. Although the system console keyboard input is ignored, all data appearing on the remote console is echoed to the system console.

6.4.5 LOCAL

The **LOCAL** door panel indicator reflects the status of the front panel keyswitch (**1 LOCAL MAINTENANCE**). However, it is not driven directly by the keyswitch. Rather, it is driven by a SCM register that has been written with the results of reading the status of the keyswitch.

When the keyswitch is set to **1 LOCAL MAINTENANCE**, the system accepts commands only from the system console. The remote console connector is ignored.

> NOTE
>
> When the front panel keyswitch is set to the **1 SECURE EXECUTION** position:
>
> - Neither the **LOCAL** nor the **REMOTE** indicator is illuminated
> - The front panel **SYSTEM RESET** is disabled
> - The remote console is disabled
> - The system attempts to automatically boot directly to multi-user CONVEX UNIX when the keyswitch is turned to this position from **0 OFF**

6.5 Power Controller Indicators

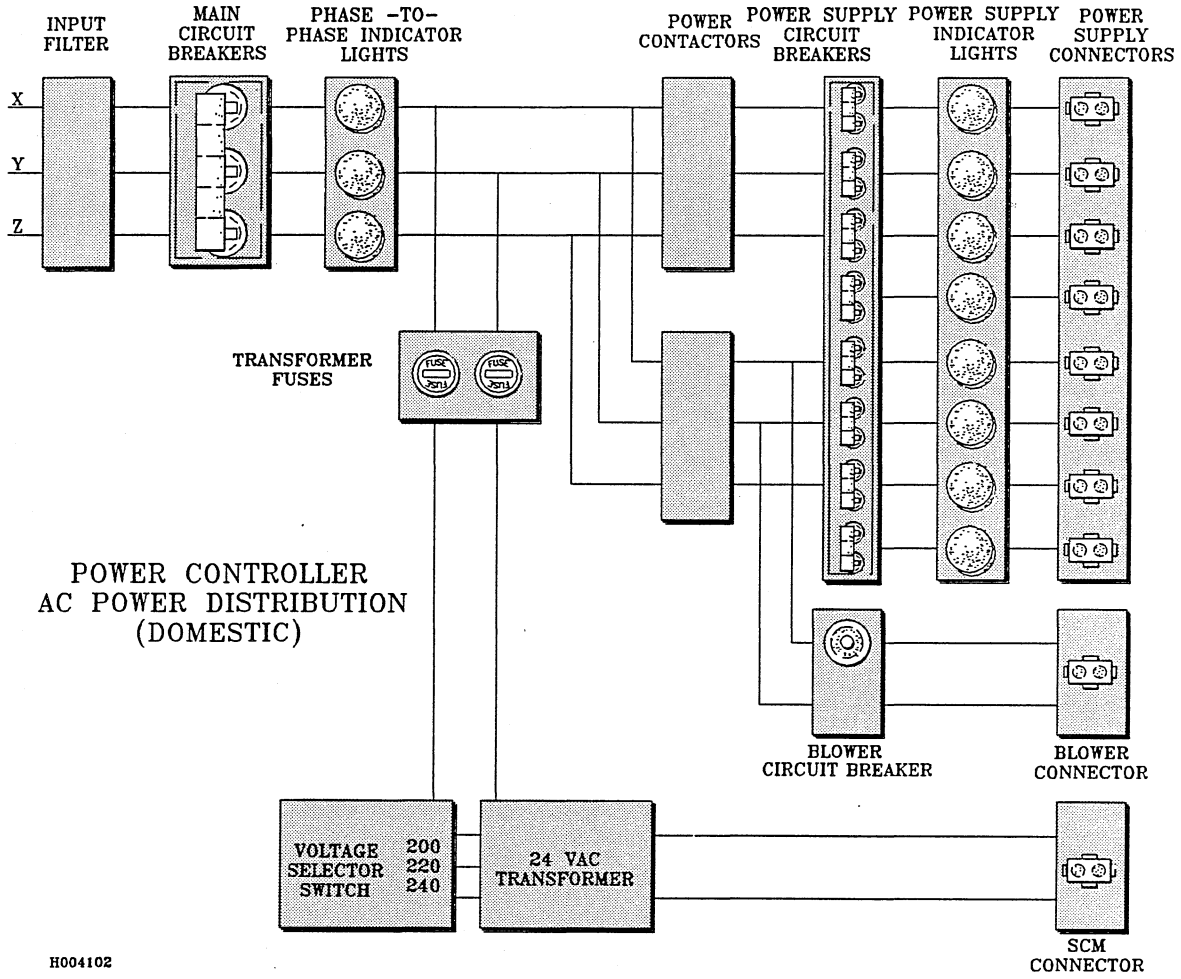
There are three phase indicator lights and eight power supply indicator lights on the power controller front panel.

The phase indicator lights show that AC power is present inside the power controller. The power supply indicator lights show which DC power supplies have power applied to their AC power input connectors on the power controller.

Since the lamps in these indicators are of the gas-discharge (neon) variety, it is unlikely that one would fail during the normal service life of the power controller.

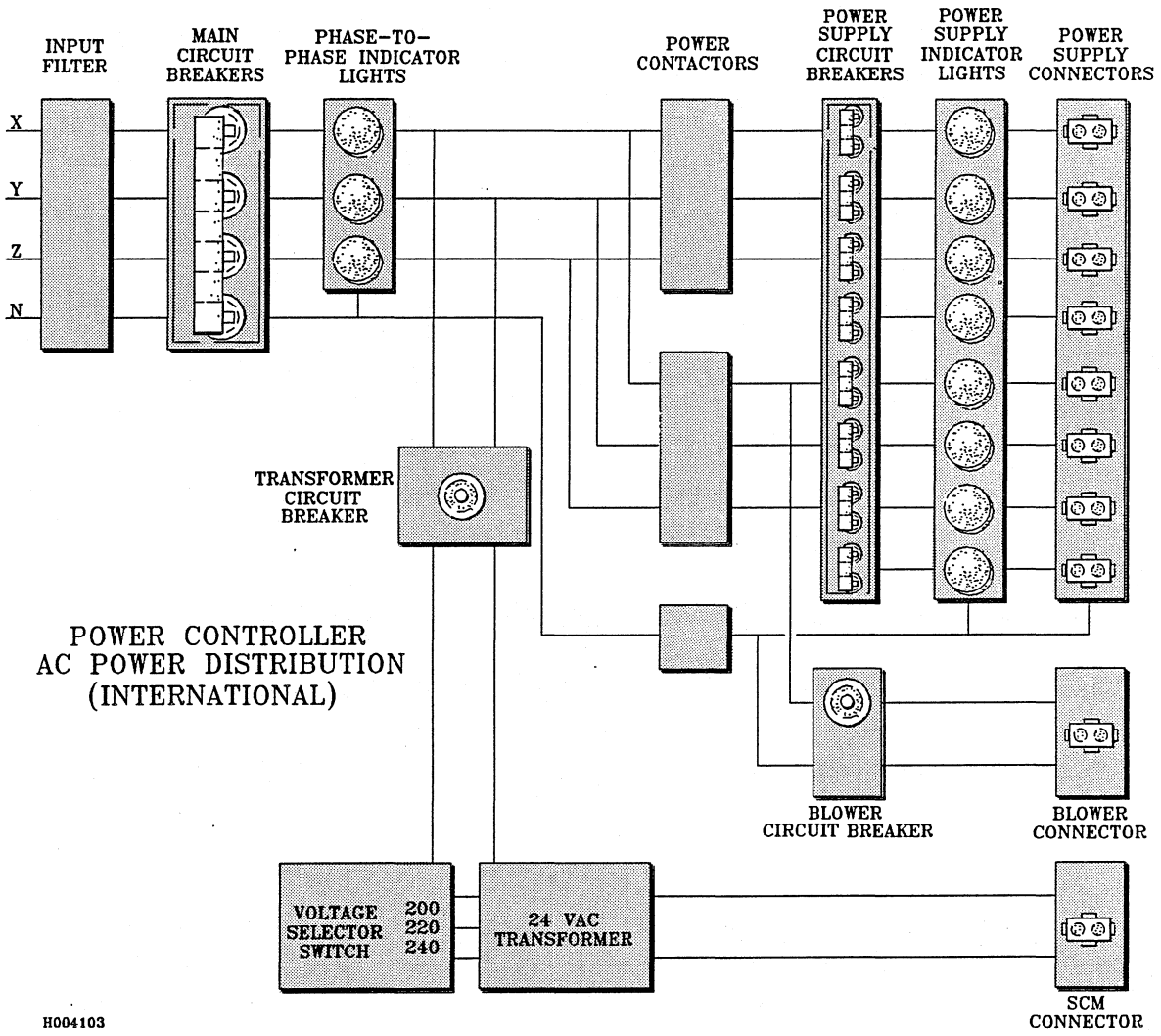
The internal AC power distribution of the domestic version of the power controller is shown in the following figure:

Figure 6–3, Power Controller AC Distribution (Domestic)



The internal AC power distribution of the international version of the power controller is shown in the following figure:

Figure 6–4, Power Controller AC Distribution (International)



6.5.1 Phase Indicator Lights

The three phase indicator lights show the presence of each of the three AC phases at the load side of the power controller main circuit breakers.

For domestic power controllers, each indicator is wired across two phases. Each indicator must have both of its phases active before it will illuminate. Therefore, loss of one phase causes the two indicators lights sharing that phase to extinguish.

For international power controllers, each indicator is wired from one phase to neutral. Loss of one phase causes only the associated indicator to extinguish.

If one phase is missing, the SCM will immediately shut the machine down and the phase indicator lights will indicate which phase is missing. The most likely cause is that one of the phases applied to the processor cabinet is missing. If power to the cabinet is good, however, the input filter, main circuit breakers, or associated wiring could be defective.

If a phase indicator light is extinguished, but the machine is still up and functioning normally, suspect the indicator light and its associated wiring.

6.5.2 Power Supply Indicator Lights

The eight power supply indicator lights show the presence of AC power at the power supply connectors on the top surface of the power controller.

> **NOTE**
>
> Only the indicator lights for the power supplies which are actually installed should be illuminated.

If a power supply indicator light that should be illuminated is not, the most likely cause is a tripped power supply circuit breaker. If the corresponding breaker is on and all phase indicator lights are also illuminated, the power supply circuit breaker or associated wiring could be defective.

If a power supply indicator light that should be illuminated is not, but the machine is still up and functioning normally, suspect the indicator light and its associated wiring.

6.6 SCM Error Codes (System Status Display)

The System Control Module (SCM) reports system status via a two-digit hexadecimal LED **SYSTEM STATUS** display on the processor cabinet front panel.

If the SCM is reporting only one condition, the corresponding code is displayed continuously. If more than one reportable condition exists, the SCM displays the code for the first condition encountered for 6 seconds, and then the code for each of the other reportable conditions (up to a total of five) for 2 seconds each, and then repeats the cycle.

NOTE

FF indicates that all measured parameters are nominal.

It is possible for the SCM to be hung in such a way that the system is down, the indicators are meaningless, and front panel switches (**RESET**) are ignored. Always try to clear this situation by power cycling the system before performing additional troubleshooting procedures:

1. Set the front panel keyswitch to **0 OFF**
2. Set the power controller main circuit breakers to **OFF**
3. Wait 10 seconds for the SCM filter capacitors to drain down
4. Set the power controller main circuit breakers to **ON**

6.6.1 SCM Error Codes — Quick Reference

The following table lists each of the SCM error codes that could appear on the front panel **SYSTEM STATUS** display along with a brief description of the error condition and its source.

Table 6–2, SCM Error Codes — Quick Reference

Code	Error Condition	Source
00	Deadman timer stop	SCM
02	SP2 interrupt acknowledge not returned	SP2 or SCM
04	SCM to SP2 data bus failure	SP2 or SCM
07	SP2 primary voltage failure	SP2 or SCM
08	A/D converter timed out	SCM
0B	Invalid SP2 command code	SP2
10	Incorrect card installed	ME0 slot
11	Incorrect card installed	MO0 slot
12	Incorrect card installed	ME1 slot
13	Incorrect card installed	MO1 slot
14	Incorrect card installed	ME2 slot
15	Incorrect card installed	MO2 slot
16	Incorrect card installed	ME3 slot
17	Incorrect card installed	MO3 slot
18	Incorrect card installed	CPX slot
19	Incorrect card installed	VPDA slot
1A	Incorrect card installed	VPDB slot
1B	Incorrect card installed	PIA slot
1C	Incorrect card installed	SP2 slot
1D	Incorrect card installed	SFUA slot
1E	Incorrect card installed	SFUB slot
1F	Incorrect card installed	ASPA slot
20	Incorrect card installed	ASPB slot
21	Incorrect card installed	IPPA slot
22	Incorrect card installed	IPPB slot
23	Incorrect card installed	VPCA slot
24	Incorrect card installed	VPCB slot
25	Incorrect card installed	DCUA slot
26	Incorrect card installed	DCUB slot
81	Bad AC power input reported	PS1
82	Bad AC power input reported	PS2
83	Bad AC power input reported	PS3
84	Bad AC power input reported	PS4
85	Bad AC power input reported	PS5
86	Bad AC power input reported	PS6
87	Bad AC power input reported	PS7
88	Bad AC power input reported	PS8

Table 6–2, SCM Error Codes — Quick Reference (continued)

Code	Error Condition	Source
89	+5 VDC power bad	PS2
8A	+12 VDC power bad	PS3
8B	-12 VDC power bad	PS3
8C	-5 VDC power bad	PS3
8D	-4.5 VDC power bad	PS5, 6, 7, 8
8E	-2 VDC power bad	PS1 or 4
91	Current load share	PS1
94	Current load share	PS4
95	Current load share	PS5
96	Current load share	PS6
97	Current load share	PS7
98	Current load share	PS8
A9	+5 VDC out of tolerance	PS2
AA	+12 VDC out of tolerance	PS3
AB	-12 VDC out of tolerance	PS3
AC	-5 VDC out of tolerance	PS3
AD	-4.5 VDC out of tolerance	PS5, 6, 7, 8
AE	-2 VDC out of tolerance	PS1, 4
B0	Intake air temp high	Intake sensor
B1	Exhaust air temp high	Exhaust sensor
B4	Insufficient airflow	CPX sensor
B5	Insufficient airflow	PIA sensor
B6	Lower bay temp high	SCM sensor
C0	Insufficient DC power	SCM
F0	Fan failure reported	Fan 0
F1	Fan failure reported	Fan 1
F2	Fan failure reported	Fan 2
F3	Fan failure reported	Fan 3
F4	Fan failure reported	Fan 4
F5	Fan failure reported	Fan 5
FF	All parameters nominal	

6.6.2 SCM Error Codes — Defined

Each of the possible SCM error codes that could appear on the front panel **SYSTEM STATUS** display are listed below along with complete description, cause, and disposition information.

00 **Description of Error:** Deadman timer timed out—system shutdown.

Cause and Disposition: The SCM deadman timer was not reset before it timed out, causing power to be removed from the system. This usually indicates a fatal internal SCM error. Reset the SCM by setting the main breakers on the power controller to **OFF**, wait 10 seconds, and then set the breakers to **ON** again. If the *00* error code is displayed again, replace the SCM.

02 **Description of Error:** SP2 interrupt acknowledge not returned—system remains up.

Cause and Disposition: SP_SM.PWRINTAK failure. This indicates that there is a problem with the communication interface between the SP2 and the SCM. When the SCM powers up the machine, one of its tasks is to reset the SP2 to force it into a known state. This error code is displayed when the SCM does not receive the reset interrupt acknowledge.

If the SP2 has not been reset properly after power up, the machine may not boot correctly. If the SCM–SP2 interface fails during machine operation, the SP2 may not be able to respond to a signal from the SCM that the machine is about to be powered down and will not have time to re-sync the disk before power down.

This condition should cleared as soon as possible to prevent loss of data.

04 **Description of Error:** SCM-to-SP2 data bus failure—system remains up.

Cause and Disposition: SP_SM.SMBDATA failure. The SCM has detected a fault in the data bus between the SCM and the SP2. Commands issued by the SP2 (power supply margin, etc.) may not be received correctly by the SCM. This can be caused by cabling problems (disconnected connectors) or faults on the SCM or SP2 boards.

07 **Description of Error:** SP2 battery backup voltage failure—system shutdown.

Cause and Disposition: The SCM determined that the SP_SM.SPUDCOK signal from the SP2 was de-asserted and shut down the system power in response. This could be caused by failure of the SP2 backup battery, or if the transfer from battery power to system power fails during powerup. The SP2 or SCM could also be faulty. Replace the SP2 first.

08 **Description of Error:** A/D converter timed out—system shutdown.

Cause and Disposition: The SCM analog-to-digital (A/D) converter did not complete a conversion, impairing the SCM's ability to monitor system environment conditions. The SCM shut down the system to prevent its operation under unmonitored (and potentially hazardous) environmental conditions. A very early version of the SCM firmware would occasionally display this code as a result of a SCM power-up transient. Power cycling the SCM (power controller main breakers) will usually clear this up. If the problem persists, replace the SCM.

0B **Description of Error:** Invalid SP2 command code received—system remains up.

Cause and Disposition: The SCM failed to correctly interpret a command from the SP2.

10 **Description of Error:** Incorrect board detected in ME0 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

11 **Description of Error:** Incorrect board detected in MO0 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

12 **Description of Error:** Incorrect board detected in ME1 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

13 **Description of Error:** Incorrect board detected in MO1 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

14 **Description of Error:** Incorrect board detected in ME2 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

15 **Description of Error:** Incorrect board detected in MO2 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

16 **Description of Error:** Incorrect board detected in ME3 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

17 **Description of Error:** Incorrect board detected in MO3 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

18 **Description of Error:** Incorrect board detected in CPX slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

19 **Description of Error:** Incorrect board detected in VPD slot of processor A—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

1A **Description of Error:** Incorrect board detected in VPD slot of processor B—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

1B **Description of Error:** Incorrect board detected in PIA slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

1C **Description of Error:** Incorrect board detected in SP2 slot—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

1D **Description of Error:** Incorrect board detected in SFU slot of processor A—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

1E **Description of Error:** Incorrect board detected in SFU slot of processor B—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

1F **Description of Error:** Incorrect board detected in ASP slot of processor A—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

20 **Description of Error:** Incorrect board detected in ASP slot of processor B—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

21 **Description of Error:** Incorrect board detected in IPP slot of processor A—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

22 **Description of Error:** Incorrect board detected in IPP slot of processor B—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

23 **Description of Error:** Incorrect board detected in VPC slot of processor A—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

24 **Description of Error:** Incorrect board detected in VPC slot of processor B—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

25 **Description of Error:** Incorrect board detected in DCU slot of processor A—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

26 **Description of Error:** Incorrect board detected in DCU slot of processor B—system power-up inhibited.

Cause and Disposition: The SCM read the ID of the board installed in this slot and determined that it is not the correct board for the slot. This is most likely caused by the wrong board having been installed in this slot. Inspect the card cage to ensure that this slot, and all other slots have the correct board installed.

81 **Description of Error:** PS1 reports bad AC input power—system shut down.

Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.

If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

82 **Description of Error:** PS2 reports bad AC input power—system shut down.
Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.
If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

83 **Description of Error:** PS3 reports bad AC input power—system shut down.
Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.
If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

84 **Description of Error:** PS4 reports bad AC input power—system shut down.
Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.
If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

85 **Description of Error:** PS5 reports bad AC input power—system shut down.
Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.
If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

86 **Description of Error:** PS6 reports bad AC input power—system shut down.
Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.
If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

87 **Description of Error:** PS7 reports bad AC input power—system shut down.

Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.

If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

88 **Description of Error:** PS8 reports bad AC input power—system shut down.

Cause and Disposition: The ACPOK line to the SCM indicates that the power supply does not have adequate AC power. This condition can have the following causes: the circuit breaker on the power controller for the power supply is tripped, power supply is unplugged from the power controller, AC input voltage is out of tolerance, AC phases for the power supply are inactive, faulty components, or AC wiring to the power supply.

If this code is displayed at power up, the power supply was not able to demonstrate control of this line and the SCM aborted power up. Change the power supply.

89 **Description of Error:** +5V DC power bad—system shut down.

Cause and Disposition: The SCM finds either no +5V DC power (PS2) on the bus or a voltage level that is less than 10% of nominal. This is most likely caused by the power supply for this voltage level not coming up to value when first enabled or catastrophic failure of the power supply during operation.

8A **Description of Error:** +12V DC power bad—system shut down.

Cause and Disposition: The SCM finds either no +12V DC power (PS3) on the bus or a voltage level that is less than 10% of nominal. This is most likely caused by the power supply for this voltage level not coming up to value when first powered on, or catastrophic failure of the power supply during operation.

8B **Description of Error:** -12V DC power bad—system shut down.

Cause and Disposition: The SCM finds either no -12V DC power (PS3) on the bus or a voltage level that is less than 10% of nominal. This is most likely caused by the power supply for this voltage level not coming up to value when first powered on, or catastrophic failure of the power supply during operation.

8C **Description of Error:** -5V DC power bad—system shut down.

Cause and Disposition: The SCM finds either no -5V DC power (PS3) on the bus or a voltage level that is less than 10% of nominal. This is most likely caused by the power supply for this voltage level not coming up to value when first powered on, or catastrophic failure of the power supply during operation.

8D **Description of Error:** -4.5V DC power bad—system shut down.

Cause and Disposition: The SCM finds either no -4.5V DC power (PS5, PS6, PS7, and PS8) on the bus or a voltage level that is less than 10% of nominal. This is most likely caused by a power supply for this voltage level not coming up to value when first enabled or catastrophic failure of a power supply during operation.

8E **Description of Error:** -2V DC power bad—system shut down.

Cause and Disposition: The SCM finds either no -2V DC power (PS1 and PS4) on the bus or a voltage level that is less than 10% of nominal. This is most likely caused by a power supply for this voltage level not coming up to value when first enabled or catastrophic failure of a power supply during operation.

91 **Description of Error:** PS1 reports out of tolerance current load share — system shut down.

Cause and Disposition: The SCM determined that the current load share of PS1 (-2V) was greater than 10% less than the current load share of PS4 (-2V). Replace indicated power supply and adjust voltage on each power supply on this power bus.

94 **Description of Error:** PS4 reports out of tolerance current load share — system shut down.

Cause and Disposition: The SCM determined that the current load share of PS4 (-2V) was greater than 10% less than the current load share of PS1 (-2V). Adjust voltage on each power supply on this power bus or replace the indicated supply and then adjust voltage.

95 **Description of Error:** PS5 reports out of tolerance current load share — system shut down.

Cause and Disposition: The SCM has determined that the current load share of PS5 (-4.5V) is out of range. Adjust voltage on each power supply on this power bus or replace the indicated supply and then adjust voltage.

96 **Description of Error:** PS6 reports out of tolerance current load share — system shut down.

Cause and Disposition: The SCM has determined that the current load share of PS6 (-4.5V) is out of range. Adjust voltage on each power supply on this power bus or replace the indicated supply and then adjust voltage.

97 **Description of Error:** PS7 reports out of tolerance current load share — system shut down.

Cause and Disposition: The SCM has determined that the current load share of PS7 (-4.5V) is out of range. Adjust voltage on each power supply on this power bus or replace the indicated supply and then adjust voltage.

98 **Description of Error:** PS8 reports out of tolerance current load share — system shut down.

Cause and Disposition: The SCM has determined that the current load share of PS8 (-4.5V) is out of range. Adjust voltage on each power supply on this power bus or replace the indicated supply and then adjust voltage.

A9 **Description of Error:** +5V DC voltage level out of tolerance.
Cause and Disposition: A +5V DC power bus (PS2) voltage level greater than ±3% of nominal causes this warning message to be displayed—system remains up. A voltage level greater than ±7% of nominal causes system shut down and display of this error message. This message is usually caused by the instability or drifting of a power supply output. Adjust voltage on this supply.

AA **Description of Error:** +12V DC voltage level out of tolerance.
Cause and Disposition: +12V DC power bus (PS3) voltage level greater than ±3% of nominal causes this warning message to be displayed—system remains up. A voltage level greater than ±7% of nominal causes system shut down and display of this error message. This message is usually caused by the instability or drifting of a power supply output. Adjust voltage on this supply.

AB **Description of Error:** -12V DC voltage level out of tolerance.
Cause and Disposition: -12V DC power bus (PS3) voltage level greater than ±3% of nominal causes this warning message to be displayed—system remains up. A voltage level greater than ±7% of nominal causes system shut down and display of this error message. This message is usually caused by the instability or drifting of a power supply output. Adjust voltage on this supply.

AC **Description of Error:** -5V DC voltage level out of tolerance.
Cause and Disposition: -5V DC power bus (PS3) voltage level greater than ±3% of nominal causes this warning message to be displayed—system remains up. A voltage level greater than ±7% of nominal causes system shut down and display of this error message. This message is usually caused by the instability or drifting of a power supply output. Adjust voltage on this supply.

AD **Description of Error:** -4.5V DC voltage level out of tolerance.
Cause and Disposition: -4.5V DC power bus (PS5, PS6, PS7, and PS8) voltage level greater than ±3% of nominal causes this warning message to be displayed—system remains up. A voltage level greater than ±7% of nominal causes system shut down and display of this error message. This message is usually caused by the instability or drifting of a power supply output. Adjust voltage on all supplies on this power bus.

AE **Description of Error:** -2V DC voltage level out of tolerance.
Cause and Disposition: -2V DC power bus (PS1 and PS4) voltage level greater than ±3% of nominal causes this warning message to be displayed—system remains up. A voltage level greater than ±7% of nominal causes system shut down and display of this error message. This message is usually caused by the instability or drifting of a power supply output. Adjust voltage on all supplies on this power bus.

B0 **Description of Error:** Input (ambient) air temperature out of tolerance.

Cause and Disposition: Intake air temperature, as sensed by a thermister attached to the bottom of the card cage assembly, above 100 °F (38 °C), but below 113 °F (45 °C), causes this warning message to be displayed—system remains up. Intake air temperature in excess of 113 °F (45 °C) causes system shut down and display of this error message. Check air filters.

B1 **Description of Error:** Exhaust air temperature out of tolerance.

Cause and Disposition: Exhaust air temperature, as sensed by a thermister attached to the fan assembly, above 131 °F (55 °C), but below 144 °F (62 °C), causes this warning message to be displayed—system remains up. Exhaust air temperature in excess of 144 °F (62 °C) causes system shut down and display of this error message. Check air filters.

B4 **Description of Error:** CPX onboard sensor reports insufficient airflow—system shut down.

Cause and Disposition: The SCM shuts the system down to prevent damage from excessive heat in the card cage. This condition could be caused by clogged filters, improperly installed air dams in the card cage, or obstructed airways in the bottom of card cage. This parameter is not measured if the CPX board is not installed in the card cage.

B5 **Description of Error:** PIA onboard sensor reports insufficient airflow—system shut down.

Cause and Disposition: The SCM shut the system down to prevent damage from excessive heat in the card cage. This condition could be caused by clogged filters, improperly installed air dams in the card cage, or obstructed airways in the bottom of card cage. This parameter is not measured if the PIA board is not installed in the card cage.

B6 **Description of Error:** Lower bay air temperature out of tolerance.

Cause and Disposition: Lower bay air temperature, as sensed by a thermister mounted on the SCM board, above 131 °F (55 °C), but below 144 °F (62 °C), causes this warning message to be displayed—system remains up. Lower bay air temperature in excess of 144 °F (62 °C) causes system shut down and display of this error message. Check air filters.

C0 **Description of Error:** Insufficient DC power installed—power-up inhibited.

Cause and Disposition: The SCM has polled the boards in the card cage to determine current requirements and the PSn-SP.XIST lines to determine the number of power supplies installed in the cabinet and found the configuration to be invalid. Check to ensure that the appropriate power supplies are installed and each control/signal cable is properly plugged into its matching power supply. Install additional DC power supplies.

F0 **Description of Error:** Fan 0 sensor reports fan failure.
Cause and Disposition: Fan failure sensed by this one fan assembly sensor *ONLY* causes this warning message to be displayed—system remains up. Failure of this fan *AND* any other fan cause system shutdown. Reports of fan failure can result from actual fan failure, from obstruction of the fan, or from failure of the fan assembly sensor. Check for obstructions and for *powered* operation of each fan.

F1 **Description of Error:** Fan 1 sensor reports fan failure.
Cause and Disposition: Fan failure sensed by this one fan assembly sensor *ONLY* causes this warning message to be displayed—system remains up. Failure of this fan *AND* any other fan cause system shutdown. Reports of fan failure can result from actual fan failure, from obstruction of the fan, or from failure of the fan assembly sensor. Check for obstructions and for *powered* operation of each fan.

F2 **Description of Error:** Fan 2 sensor reports fan failure.
Cause and Disposition: Fan failure sensed by this one fan assembly sensor *ONLY* causes this warning message to be displayed—system remains up. Failure of this fan *AND* any other fan cause system shutdown. Reports of fan failure can result from actual fan failure, from obstruction of the fan, or from failure of the fan assembly sensor. Check for obstructions and for *powered* operation of each fan.

F3 **Description of Error:** Fan 3 sensor reports fan failure.
Cause and Disposition: Fan failure sensed by this one fan assembly sensor *ONLY* causes this warning message to be displayed—system remains up. Failure of this fan *AND* any other fan cause system shutdown. Reports of fan failure can result from actual fan failure, from obstruction of the fan, or from failure of the fan assembly sensor. Check for obstructions and for *powered* operation of each fan.

F4 **Description of Error:** Fan 4 sensor reports fan failure.
Cause and Disposition: Fan failure sensed by this one fan assembly sensor *ONLY* causes this warning message to be displayed—system remains up. Failure of this fan *AND* any other fan cause system shutdown. Reports of fan failure can result from actual fan failure, from obstruction of the fan, or from failure of the fan assembly sensor. Check for obstructions and for *powered* operation of each fan.

F5 **Description of Error:** Fan 5 sensor reports fan failure.
Cause and Disposition: Fan failure sensed by this one fan assembly sensor *ONLY* causes this warning message to be displayed—system remains up. Failure of this fan *AND* any other fan cause system shutdown. Reports of fan failure can result from actual fan failure, from obstruction of the fan, or from failure of the fan assembly sensor. Check for obstructions and for *powered* operation of each fan.

FF **Description of Indication:** All measured parameters are nominal.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7
Hard Error Messages

7.1 Overview

This chapter lists the various hard error messages with a definition, cause and disposition, data flow diagram, and list of likely failed boards for each.

7.2 Hard Error Messages

Hard errors are usually parity errors detected in registers and on buses. They are arranged according to their source and assigned a three-digit error number:

- ASP (100 series)
- CPX (400 series)
- DCU (200 series)
- IPP (300 series)
- MCM (500 series)
- PIA (600 series)
- VPC (700 series)
- VPD (800 series)

Each of these categories is covered in a separate section.

7.2.1 ASP Hard Errors

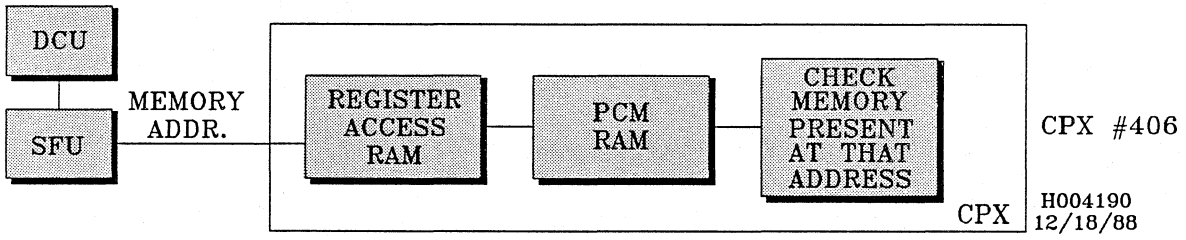
The following messages are displayed when a hard error is reported by the ASP board.

> **NOTE**
>
> x denotes CPU (A, B, C, or D)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

`ASPx/ASP: [#100] CBUS parity error: syndrome = %02x`

Description of Error: Parity checker detected error in DCACHE output to register.

Figure 7–1, ASP #100 data flow



Cause and Disposition: This can be caused by bad data or parity from the MCM or the VPD, by problems on the ASP, or by bus problems on the backplane. Run diagnostics *cpu_4231* subtests 510–519 and *cpu_4041* subtests 200–210, 550–566 to check CPU functions, and then *mem_4000* subtests 401, 300–302 to check the MCM.

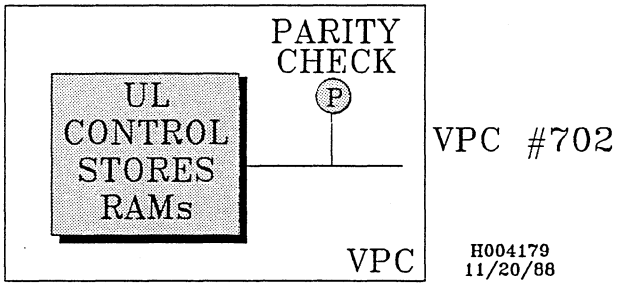
Most likely failed boards (listed alphabetically):

- **ASP**—Contains DCACHE CBUS and register file
- **Backplane**—Contains data paths
- **DCU**—Source of address and write enable
- **MCM**—Data from memory can come from any MCM
- **VPD**—DCACHE data comes from this board

`ASPx/ASP: [#101] DBUS parity error: syndrome = %02x`

Description of Error: Parity checker detected error in data input to register.

Figure 7–2, ASP #101 data flow



Cause and Disposition: This can be caused by bad data or parity from the MCM or the VPD, by problems on the ASP, or by bus problems on the backplane. Run diagnostics *cpu_4231* subtests 510–519 and *cpu_4041* subtests 200–216, 230–246 to check CPU functions, and then *mem_4000* subtests 401, 300–302 to check the MCM.

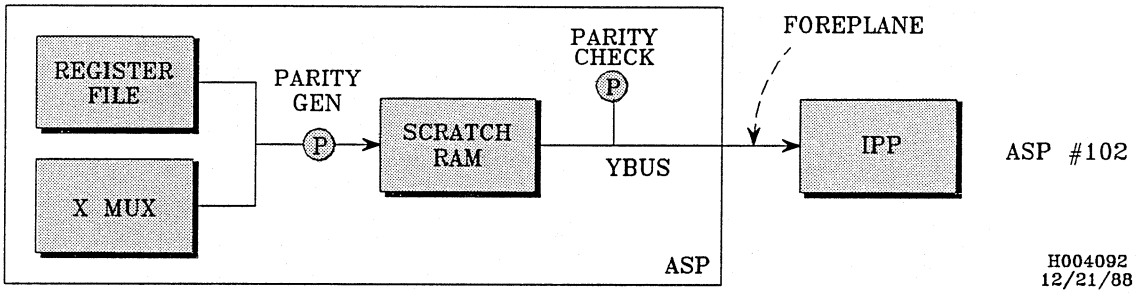
Most likely failed boards (listed alphabetically):

- **ASP**—DBUS register file and source selection
- **Backplane**—Contains data paths
- **MCM**—Data from memory can come from any MCM
- **VPD**—Register data comes from this board

`ASPx/ASP: [#102] SRAM parity error: syndrome = %02x`

Description of Error: Parity checker detected error in output from scratch RAM.

Figure 7–3, ASP #102 data flow



Cause and Disposition: This can be caused by problems with the register file, XMUX, on the XBUS or the YBUS on the ASP, by bus problems on the backplane, or by problems on the IPP. Run diagnostics *cpu_4030*, *cpu_4232*, and then *cpu_4231* to check CPU functions.

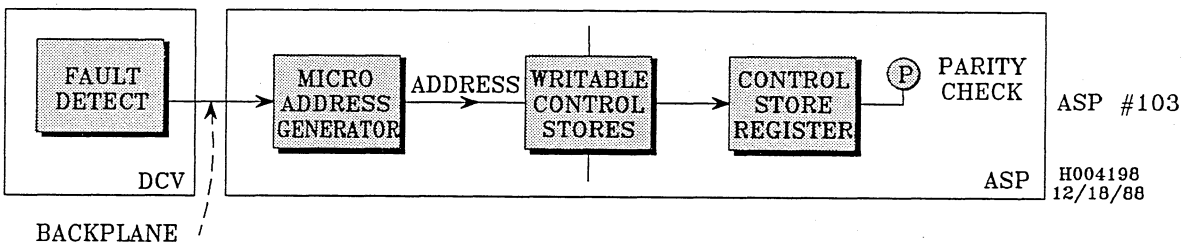
Most likely failed boards (listed alphabetically):

- **ASP**—Location of scratch ram
- **Foreplane**—Contains YBUS data paths to IPP
- **IPP**—This card also sits on the YBUS

`ASPx/ASP: [#103] WCS parity error: (upc - 1) = %03x`

Description of Error: Parity checker detected error in writable control store output. Parity information is scanned *into* the control store.

Figure 7–4, ASP #103 data flow



Cause and Disposition: This can be caused by problems with the writable control store logic on the ASP, by problems with the fault detect logic on the DCU, or by problems on the backplane. Run the control store loader utility with the verify option set to **ON** (**cs -v**) to verify the existance of a writable control store parity error.

Most likely failed boards (listed alphabetically):

- **ASP**—Contains writable control store and parity checker
- **Backplane**—Contains data paths
- **DCU**—Generates fault signal to microaddress generator

`ASPx/ASP: [#104] microcode error: ipc = %03x`

Description of Error: ASP microcode detected an unrecoverable error. Error message indicates likely source (refer to [text string] at *Most likely failed boards*, below).

Cause and Disposition: This can be caused by an invalid fault vector from DCU fault detect logic, by bad data or parity from memory, by a bad dispatch from the IPP, by an ASP microsequencer error, or by bus problems on the backplane. Run diagnostics *cpu_4231* subtests 10–36 in the default mode first, and then with forced faults and faulting on IP fetchs enabled.

Most likely failed boards (listed alphabetically):

- **ASP**—Contains microsequencer [*any*]
- **Backplane**—Contains data paths
- **DCU**—Source of fault vectors [BF4, BF6, BF8, or BFF]
- **IPP**—Source of dispatch [BFC]
- **MCM**—Source of memory data and parity [BF6, BF8, BFF]

7.2.2 CPX Hard Errors

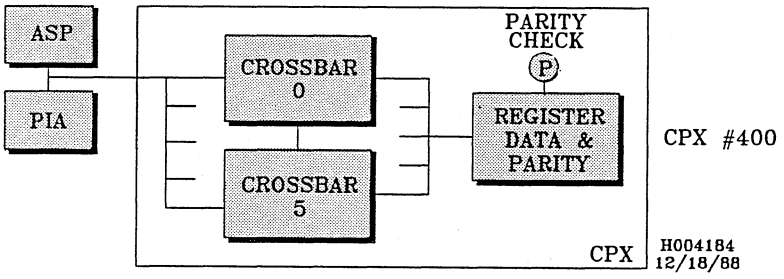
The following messages are displayed when a hard error is reported by the CPX board.

> **NOTE**
>
> x denotes CPU (A, B, C, or D)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

`CPX: [#400] Post X-bar parity error. X-bar data: %04x, Parity: %0x`

Description of Error: Parity checker detected error in data output from the crossbars.

Figure 7–5, CPX #400 data flow



Cause and Disposition: This can be caused by bad data or parity from the ASP or the PIA, by problems on the CPX, or by bus problems on the backplane. Run diagnostics *cpx_4000* subtests 130 and 135 to check CPX functions, and then *cpu_4232* subtests 110–135 to check the ASP, and *io_4000* to check the PIA.

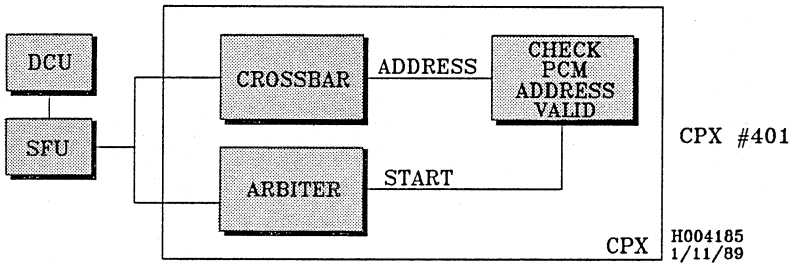
Most likely failed boards (listed alphabetically):

- **ASP**—Source of data and parity
- **Backplane**—Contains data paths
- **CPX**—Contains crossbar gate arrays
- **PIA**—Source of data and parity

```
CPX: [#401] Invalid address for PCM request (pcm_start_err)
CPX: PCM address<25..17> (I/O addr<28..20>): %03x
```

Description of Error: An attempt was made to access an unimplemented address in PCM space.

Figure 7–6, CPX #401 data flow



Cause and Disposition: This can be caused by a problem on the DCU (generates the address), on the SFU (sends the address), on the CPX (decodes the address and checks for errors), or with the address buses. Run diagnostics *cpx_4000* subtests 180 and 185 to check CPX functions, and then *cpu_4010* subtests 510–620 to verify that valid PCM addresses are not being translated into invalid addresses.

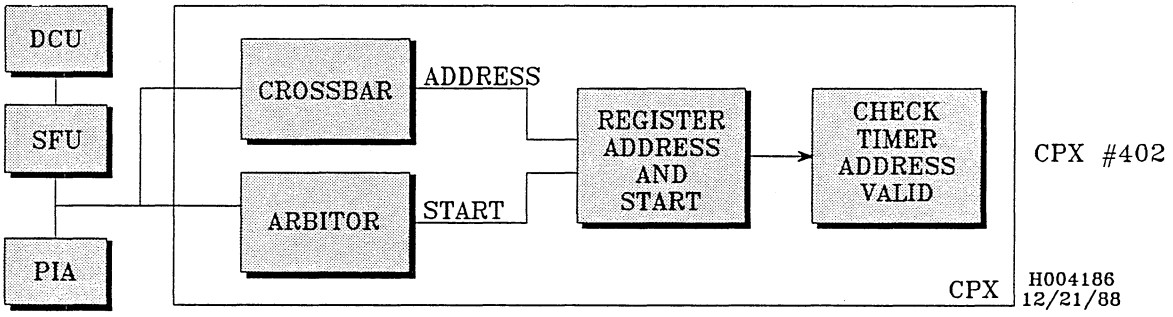
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains address path
- **CPX**—Contains address decode and checking
- **DCU**—Generates address
- **SFU**—Sends address

`CPX: [#402] Timer/Counter address error`

Description of Error: An attempt was made to access an unimplemented address within the timer/counter space.

Figure 7–7, CPX #402 data flow



Cause and Disposition: This can be caused by a problem on the DCU which generates the address, on the SFU or PIA (send the address), on the CPX (decodes the address and checks for errors), or with the address buses. Run diagnostics *cpx_4000* subtests 140 and 180 to check CPX functions, and then *cpu_4231* subtests 41 and 604 to verify that valid addresses are being translated correctly.

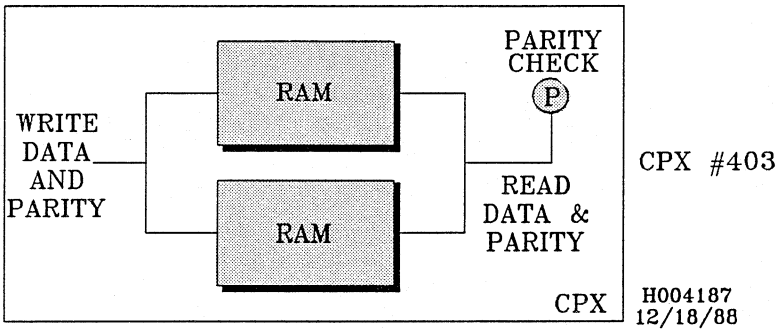
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains address path
- **CPX**—Contains address decode and checking
- **DCU**—Generates address
- **PIA**—Sends address
- **SFU**—Sends address

`CPX: [#403] Timer/Counter RAM parity error`

Description of Error: Parity checker detected error in timer/counter RAM.

Figure 7–8, CPX #403 data flow



Cause and Disposition: This is caused by problems with the counter/timer RAMs on the CPX. Run diagnostics *cpx_4000* subtests 220 and 225 to check CPX counter/timer functions.

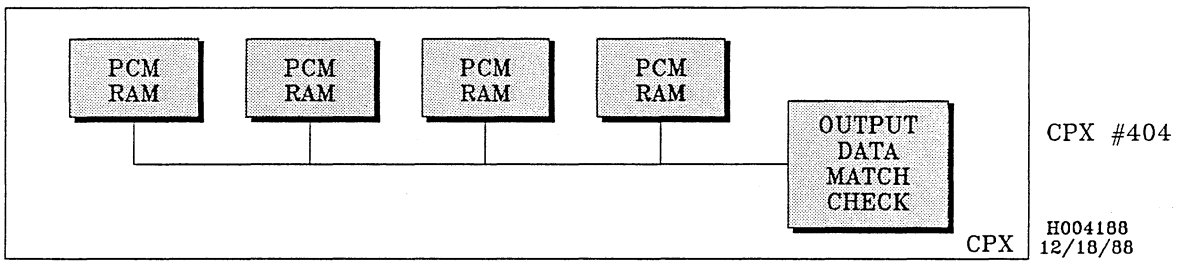
Most likely failed board:

- **CPX**—Contains counter/timer RAMs

`CPX: [#404] PCM RAM error: banks contain inconsistent data`

Description of Error: PCM RAMs were found to have inconsistent data—all PCM RAM data should match.

Figure 7–9, CPX #404 data flow



Cause and Disposition: This is caused by problems in the PCM RAM area of the CPX. Run diagnostics *cpx_4000* subtest 205 to check CPX PCM RAMs.

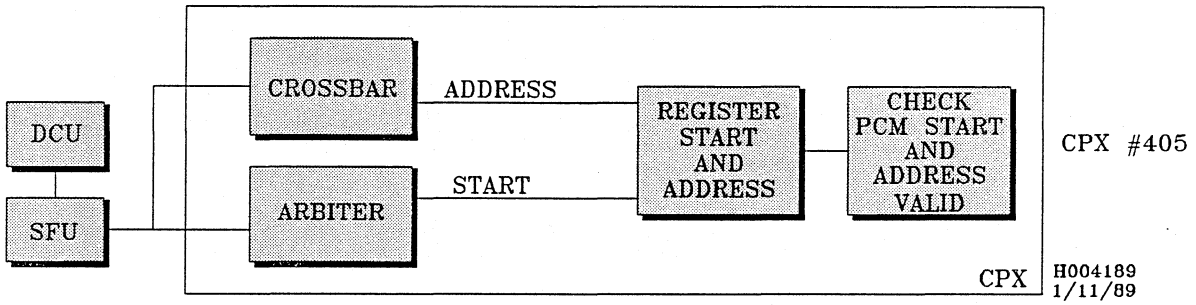
Most likely failed board:

- **CPX**—Contains PCM RAMs

`CPX: [#405] Invalid address for PCM request`

Description of Error: An attempt was made to access an unimplemented address in PCM space.

Figure 7–10, CPX #405 data flow



Cause and Disposition: This can be caused by a problem on the DCU (generates the address), on the SFU (sends the address), on the CPX (decodes the address and checks for errors), or with the address buses. Run diagnostics *cpx_4000* subtests 157 and 180 to check CPX functions, and then *cpu_4010* subtests 510–620 to check the CPU.

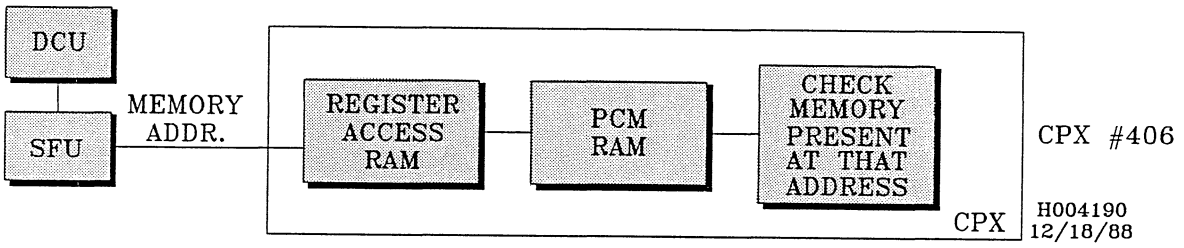
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains address path
- **CPX**—Contains address decode and checking
- **DCU**—Generates address
- **SFU**—Sends address

`CPX: [#406] PCM memory error: Request to nonexistent memory`

Description of Error: An attempt was made to access system memory that is not installed.

Figure 7–11, CPX #406 data flow



Cause and Disposition: This can be caused by a problem on the DCU (generates the address), on the SFU (sends the address), on the CPX (decodes the address and checks for errors), or with the address buses. Run diagnostics *cpx_4000* subtest 185 to check CPX functions, and then *cpu_4030* subtests 200–555 and *cpu_4231* subtests 10–36 to check the CPU.

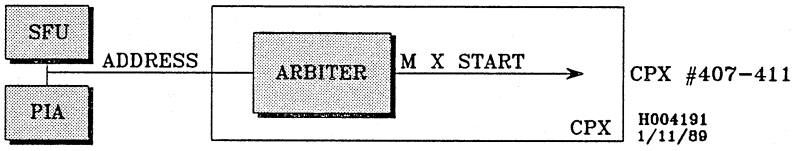
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains address path
- **CPX**—Contains address decode and checking
- **DCU**—Generates address
- **SFU**—Sends address

```
CPX: [#407] Illegal access to I/O space: m2start
CPX: [#408] Illegal access to I/O space: m4start
CPX: [#409] Illegal access to I/O space: m5start
CPX: [#410] Illegal access to I/O space: m6start
CPX: [#411] Illegal access to I/O space: m7start
```

Description of Error: An attempt was made to access unimplemented I/O space.

Figure 7–12, CPX #407–#411 data flow



Cause and Disposition: This can be caused by a problem on the DCU (generates the address), the SFU or PIA (send the address), on the CPX (decodes the address and checks for errors), or with the address buses. Run diagnostics *cpx_4000* subtest 180 to check CPX functions, and then *cpu_4010* subtests 10, 20, and 510–620 to check the CPU.

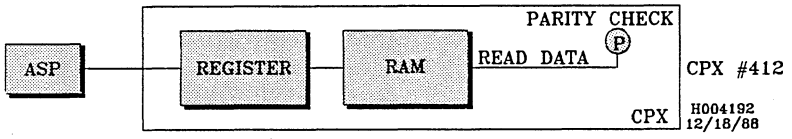
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains address path
- **CPX**—Contains address decode and checking logic
- **DCU**—Generates address
- **PIA**—Sends address
- **SFU**—Sends address

```
CPX: [#412] Comm register parity error. Addr <9..0> = %03x
```

Description of Error: Parity checker detected error in data read from comm register RAMs.

Figure 7–13, CPX #412 data flow



Cause and Disposition: This can be caused by problems on the ASP (generates the data), with the comm register RAMs on the CPX, or with the data paths. Run diagnostics *cpx_4000* subtests 195 and 215 to check CPX functions, and then *cpu_4232* subtests 110–125 to check the CPU.

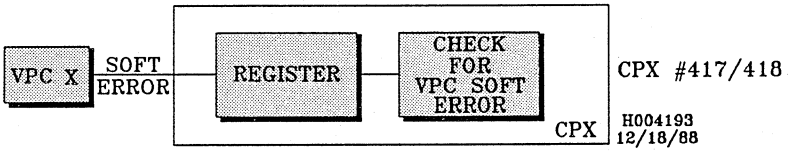
Most likely failed boards (listed alphabetically):

- **ASP**—Generates data
- **Backplane**—Contains data paths
- **CPX**—Contains comm register RAMs

```
CPX: [#417] VPCA soft error detected
CPX: [#418] VPCB soft error detected
```

Description of Error: A soft error has been reported by a VPC.

Figure 7–14, CPX #417/#418 data flow



Cause and Disposition: This is most likely caused by a parity error detected in the VPC scan ring, a problem with the CPX hardware that registers the error report, or with the connecting data paths. Run diagnostics *cpx_4000* subtest 160 to check CPX functions, and then *cpu_4041* (all tests) to check the VPC.

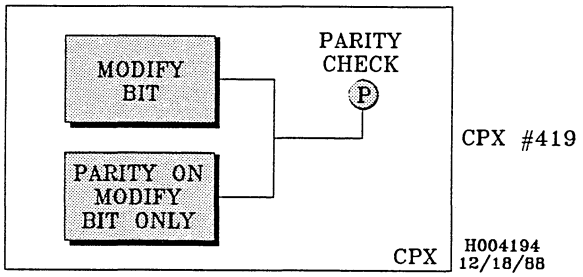
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains data paths
- **CPX**—Registers soft error report
- **VPC**—Contains scan ring and error detection logic

```
CPX: [#419] R/M parity error
rm_addr <18..0>: %08x
Bank 0 modified bit: %x, Parity: %x
Bank 1 modified bit: %x, Parity: %x
```

Description of Error: Parity checker detected error in data read from modified RAM.

Figure 7–15, CPX #419 data flow



Cause and Disposition: This is most likely caused by problems with the R&M hardware on the CPX. Run diagnostics *cpx_4000* subtest 210 to check CPX functions.

Most likely failed board:

- **CPX**—Contains R&M hardware

```
CPX: [#420] PCM parity error
```

Then one of the following:

```
DCUA even memory request
DCUA odd memory request
DCUB even memory request
DCUB odd memory request
```

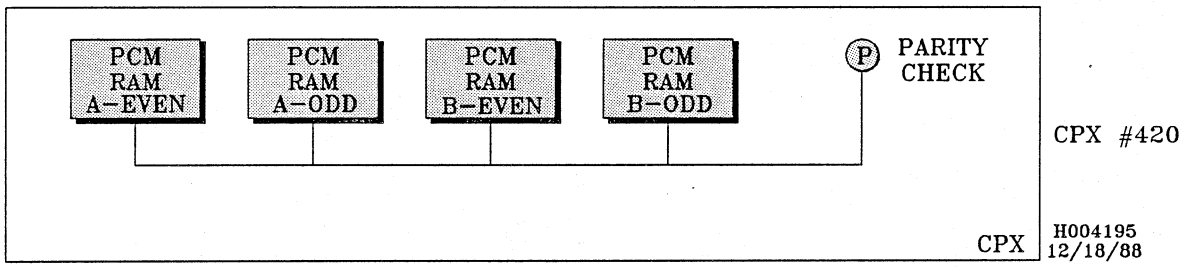
I/O request **All addresses should be the same

Then the following:

```
CPUA even mbus PCM address: %08x, Data: %08x, Parity: %02x
CPUA odd mbus PCM address: %08x, Data: %08x, Parity: %02x
CPUB even mbus PCM address: %08x, Data: %08x, Parity: %02x
CPUB odd mbus PCM address: %08x, Data: %08x, Parity: %02x
Resultant PCM data: %08x parity: %02x
```

Description of Error: Parity checker detected error in data read from PCM RAMs.

Figure 7–16, CPX #420 data flow



Cause and Disposition: This is most likely caused by bad data being written into PCM RAM or hardware problems with the PCM RAMs. Run diagnostics *cpx_4000* subtest 205 to check CPX functions.

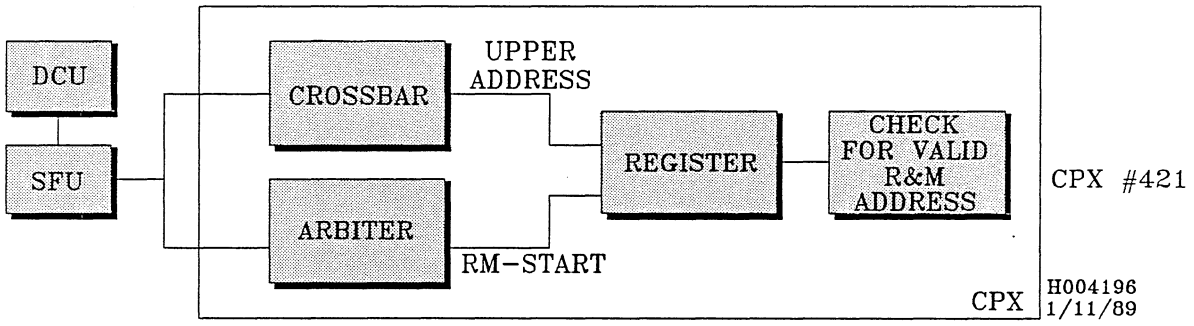
Most likely failed board:

- **CPX**—Contains PCM RAMs

```
CPX: [#421] "Invalid address for R/M request (rm_start_err)
CPX: R/M address<25..17> (I/O addr<28..20>): %03x
```

Description of Error: A read or write operation has been directed to an invalid address in the R&M space.

Figure 7–17, CPX #421 data flow



Cause and Disposition: This can be caused by a problem on the DCU (generates the address), on the SFU (sends the address), on the CPX (decodes the address and checks for errors), or with the address buses. Run diagnostics *cpx_4000* subtest 180 to check CPX functions, and then *cpu_4010* subtests 10–36 to check the CPU.

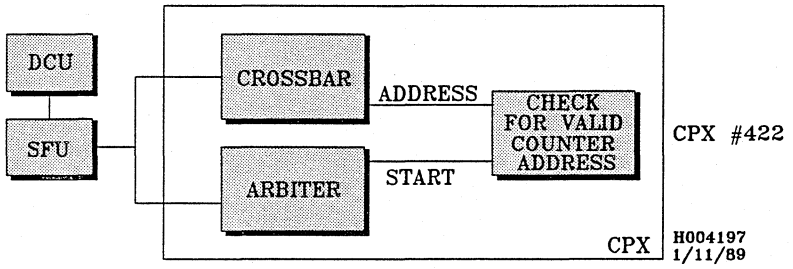
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains address path
- **CPX**—Contains address decode and checking logic
- **DCU**—Generates address
- **SFU**—Sends address

```
CPX: [#422] Invalid address for counter request (ti_start_err)
CPX: counter address<25..17> (I/O addr<28..20>): %03x
```

Description of Error: A read or write operation has been directed to an invalid address in the upper 9 bits of the counter address.

Figure 7–18, CPX #422 data flow



Cause and Disposition: This can be caused by a problem on the DCU (generates the address), on the SFU or PIA (send the address), on the CPX (decodes the address and checks for errors), or with the address buses. Run diagnostics *cpx_4000* subtest 170 to check CPX functions, and then *cpu_4032* subtests 110–125 and *cpu_4231* subtest 41 to check the CPU.

Most likely failed boards (listed alphabetically):

- **Backplane**—Contains address path
- **CPX**—Contains address decode and checking logic
- **DCU**—Generates address
- **PIA**—Sends address
- **SFU**—Sends address

7.2.3 DCU Hard Errors

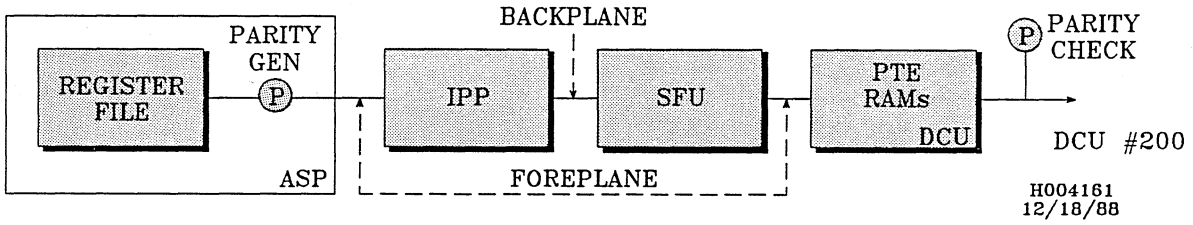
The following messages are displayed when a hard error is reported by the DCU board.

> **NOTE**
>
> x denotes CPU (A, B, C, or D)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

`DCUx/DCU: [#200] PTE Cache parity error`

Description of Error: Parity checker detected error in PTE cache RAMs on DCU.

Figure 7–19, DCU #200 data flow



Cause and Disposition: This error usually indicates a backplane problem (the bus that writes the PTE cache crosses all 4 scalar boards via the foreplanes and backplane). Run diagnostic *cpu_4231* subtest 400 to tests all PTE cache RAM cells (except cell 1023, which is associated with the execution of the test).

Most likely failed boards (listed alphabetically):

- **ASP, IPP, or SFU**—Bus that writes PTE cache generated or buffered on these boards
- **Backplane**—Problem likely on the IP-FU.ZCX bus
- **DCU**—Control and addressing logic of PTE cache RAMs
- **Foreplane**—Failure on the AS-IP.Z bus (ASP–IPP foreplane) or on the FU-DC.ZCX bus (DCU–SFU foreplane).

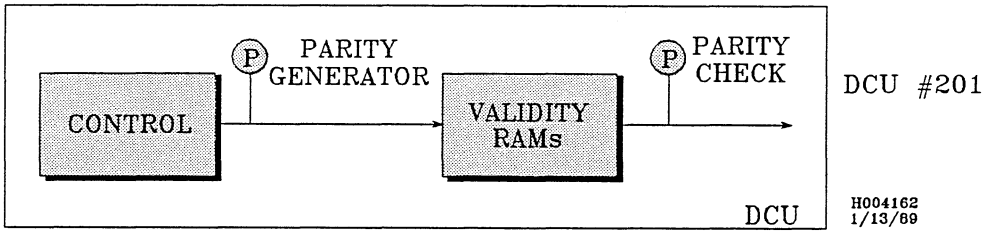
NOTE

If simultaneous PTE cache error (DCU #200) and data cache parity errors (DCU #201–DCU #205) are reported, it is likely that cold start is failing. This indicates an ASP problem.

`DCUx/DCU: [#201] Data Cache validity parity error`

Description of Error: Parity checker detected error in data cache validity RAMs on DCU.

Figure 7–20, DCU #201 data flow



Cause and Disposition: This is most likely caused by failed validity RAMs on the DCU. Run diagnostic *cpu_4231* subtests 510–519 to test all data cache validity RAM cells.

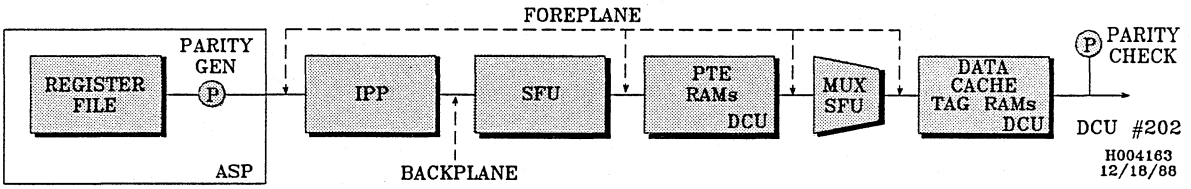
Most likely failed board:

- **DCU**—Contains all validity RAMs and control logic

`DCUx/DCU: [#202] Data Cache Tag parity error`

Description of Error: Parity checker detected error in data cache tag RAMs on DCU.

Figure 7–21, DCU #202 data flow



Cause and Disposition: This could be caused by bad data from PTE cache RAMs, by problems with the data cache tag RAMs on the DCU, by problems on the SFU, or by problems with the data path. If this error is reported along with PTE cache error (DCU #200), refer to DCU #200 first. Run diagnostic *cpu_4231* subtests 510–519 to test all data cache tag RAM cells.

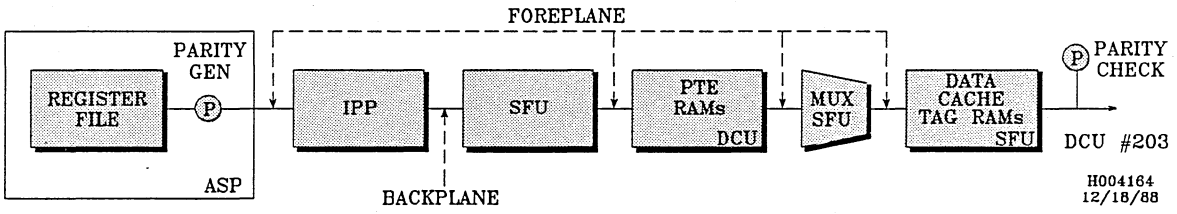
Most likely failed boards (listed alphabetically):

- **DCU**—Contains PTE cache RAMS, data cache tag RAMs, and associated control logic
- **Foreplane**—Two foreplane transitions in write data path for tag RAMs
- **SFU**—Multiplexes write data

`DCUx/DCU: [#203] Parity error in remote validate RAMs`

Description of Error: Parity checker detected error in data cache tag RAMs on *SFU* board.

Figure 7–22, DCU #203 data flow



Cause and Disposition: This could be caused by bad data from PTE cache RAMs on the DCU, by problems with the data cache tag RAMs on the DCU, by problems on the SFU, or by problems with the data path. If this error is reported along with PTE cache error (DCU #200), refer to DCU #200 first. Run diagnostic *cpu_4231* subtest 560–570 in a single-CPU environment, and subtest 500–571 in a multi-CPU environment to test all remote invalidate RAM cells.

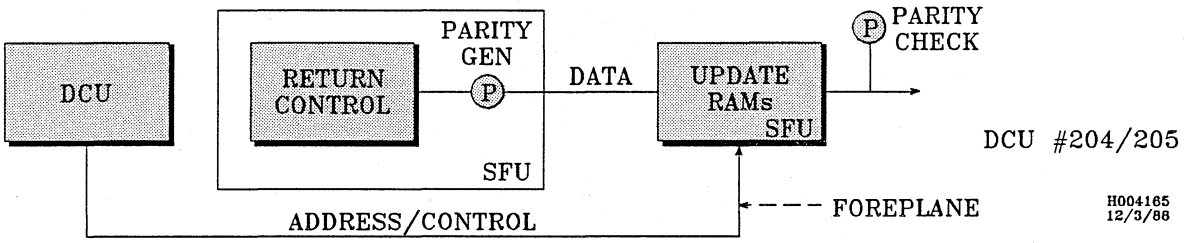
Most likely failed boards (listed alphabetically):

- **DCU**—Controls remote invalidate RAMs
- **Foreplane**—DCU–SFU connection
- **SFU**—Multiplexes write data for tag RAMs, contains remote invalidate RAMs

```
DCUx/DCU: [#204] Error in even side update tag RAMs
DCUx/DCU: [#205] Error in odd side update tag RAMs
```

Description of Error: Parity checker detected error in data cache update RAMs on *SFU* board.

Figure 7–23, DCU #204/205 data flow



Cause and Disposition: This could be caused by problems on the DCU, by problems on the SFU, or by problems with the foreplane data paths. Run diagnostic *cpu_4231* subtests 510–569 to test all cache update RAM cells.

Most likely failed boards (listed alphabetically):

- **DCU**—Generates all address and control
- **Foreplane**—DCU–SFU connection
- **SFU**—Contains RAMs and generates data

7.2.4 IPP Hard Errors

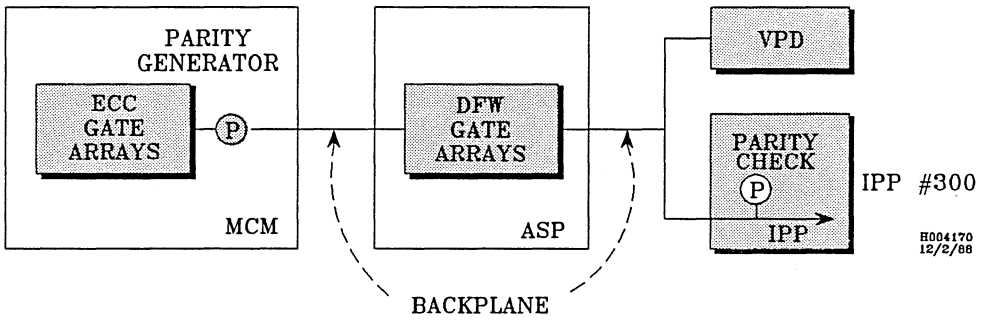
The following messages are displayed when a hard error is reported by the IPP board.

> **NOTE**
>
> x denotes CPU (A, B, C, or D)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

`IPPx/IPP: [#300] mbus_par_err = %02x`

Description of Error: This message indicates that a parity error was detected on the AS-BP.DATA <63..0> bus. This bus is registered by the IPP on every clock.

Figure 7–24, IPP #300 data flow



Cause and Disposition: Most likely causes: Problems with the DFW gate arrays on the ASP or bad data from the MCM. The probability of a memory problem increases as the number of MCM pairs installed increases. This error could also be caused by failed bus terminations on the VPD or backplane bus problems (ASP to IPP or MCM to ASP). Run *mminit*, diagnostic *mem_4000* subtests 401 and 350 to check the memory system, then *cpu_4030* subtests 200–555 to check the ASP DFW gate arrays.

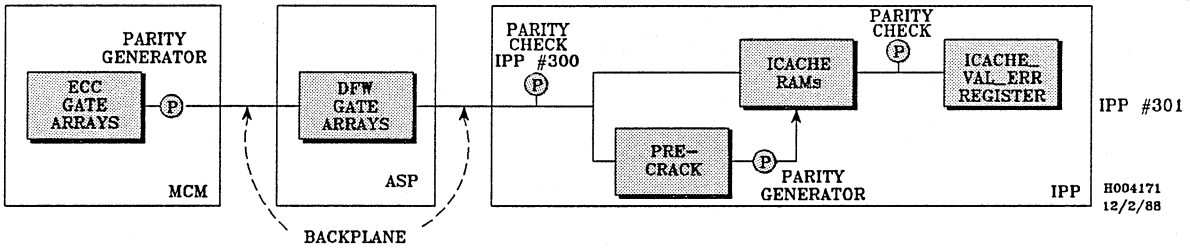
Most likely failed boards (listed alphabetically):

- **ASP**—Contains DFW gate arrays
- **Backplane**—Contains data paths
- **MCM**—Source of data to DFW gate arrays
- **VPD**—Contains MBUS terminations

`IPPx/IPP: [#301] icache validity bad. icache_val_err = %01x`

Description of Error: Parity checker detected error while reading ICACHE validity register bits.

Figure 7–25, IPP #301 data flow



Cause and Disposition: Most likely causes: Problems with the ICACHE RAMs on IPP or with data path from ICACHE.

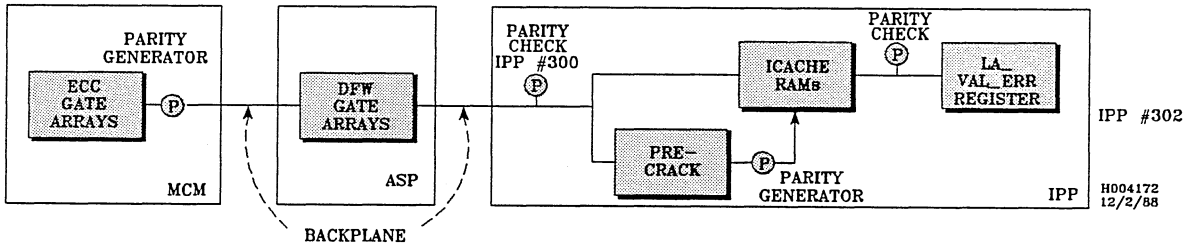
Most likely failed board:

- **IPP**—Board on which ICACHE RAMs and ICACHE validity registers are located

`IPPx/IPP: [#302] lookahead validity bad. la_val_err = %01x`

Description of Error: Parity checker detected error while reading ICACHE lookahead register bits.

Figure 7–26, IPP #302 data flow



Cause and Disposition: Problems with the ICACHE RAMs on IPP or with data path from ICACHE.

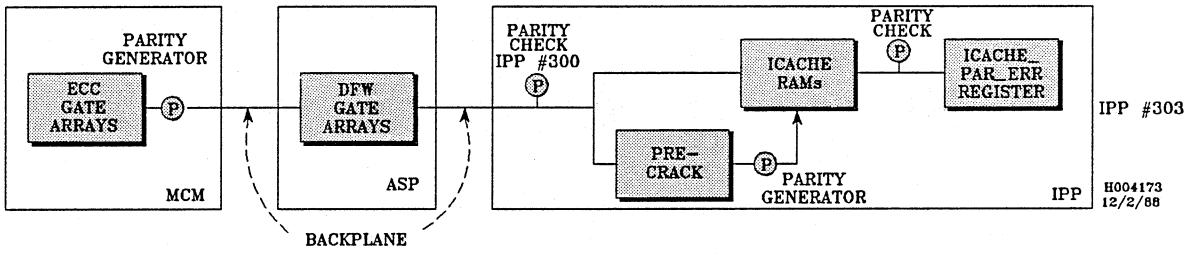
Most likely failed board:

- **IPP**—Board on which ICACHE RAMs and lookahead register are located

`IPPx/IPP: [#303] icache_par_err = %04x`

Description of Error: Parity checker detected error in ICACHE output data.

Figure 7–27, IPP #303 data flow



Cause and Disposition: Problems with the ICACHE RAMs on IPP or with data path from ICACHE.

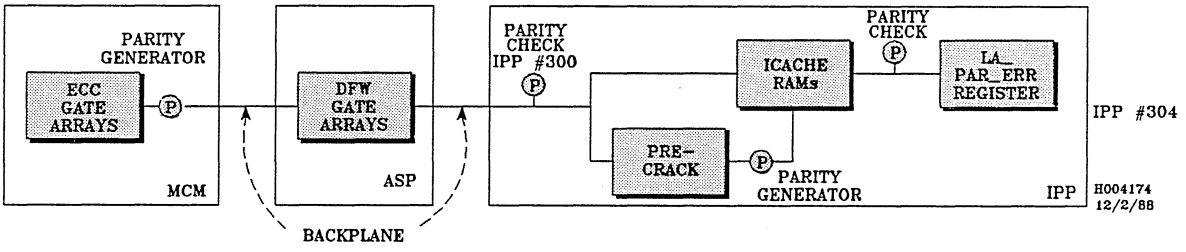
Most likely failed board:

- **IPP**—Board on which ICACHE RAMs are located

```
IPPx/IPP: [#304] la_par_err = %01x
  IPPx/IPP: l.la_tag<31..24> bad
  IPPx/IPP: l.la_tag<23..16> bad
  IPPx/IPP: l.la_tag<15..13> bad
```

Description of Error: Parity checker detected error when lookahead tag is read.

Figure 7–28, IPP #304 data flow



Cause and Disposition: Problems with the ICACHE RAMs on IPP or with data path from ICACHE.

Most likely failed board:

- **IPP**—Board on which ICACHE RAMs and lookahead tag register is located

7.2.5 MCM Hard Errors

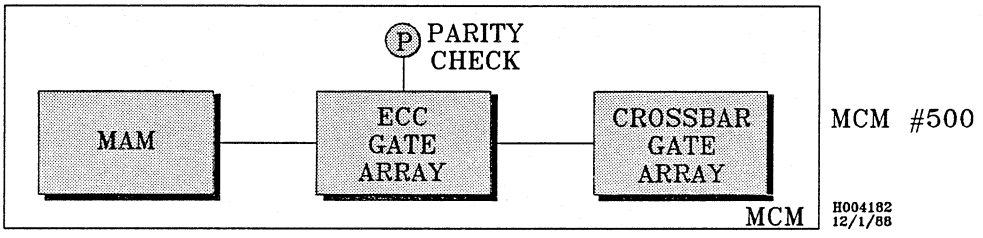
The following messages are displayed when a hard error is reported by the MCM board.

> **NOTE**
>
> *x* indicates MCM pair (0, 1, 2, or 3)
>
> *y* denotes which MCM of the pair (E or O)
>
> *unit d%* indicates which bank failed (0–7)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

```
MCMxy/MCM: [#500] ECC unit %d: multiple bit error detected
MCMxy/MCM: Interleaved address: %08X, data: %08x
MCMxy/MCM: Actual ECC: %02X, Generated ECC: %02X
```

Description of Error: Main memory system (MCM) detected multiple bit ECC error from a MAM. Data and ECC are stored together in a memory location. When a memory location is read, new ECC is generated from the data from memory and then compared to the ECC from memory. If this compare fails, a hard error is generated.

Figure 7–29, MCM #500 data flow



Cause and Disposition: The ECC gate array has detected invalid ECC on a read (or a partial write) of memory. This can be caused by uninitialized memory, a problem in the ECC gate arrays, problems with the WR_ECC, RD_ECC, or RAM_DAT buses on the MCM, or a failed MAM.

Run diagnostics *mem_4000* subtests 550–560, and 170 to pattern-test the memory arrays, then initialize the memory with *mminit*, and run processor diagnostics *cpu_4041* subtests 200–216, and then 550–566 to check for crosstalk and other problems with data sources.

> **NOTE**
>
> Always run *mminit* to initialize memory prior to running any diagnostic other than *mem_4000* and prior to booting the system.
>
> A level 2 reset (*sysreset* level 2) to the memory boards will destroy data and **un**initialize the memory system.

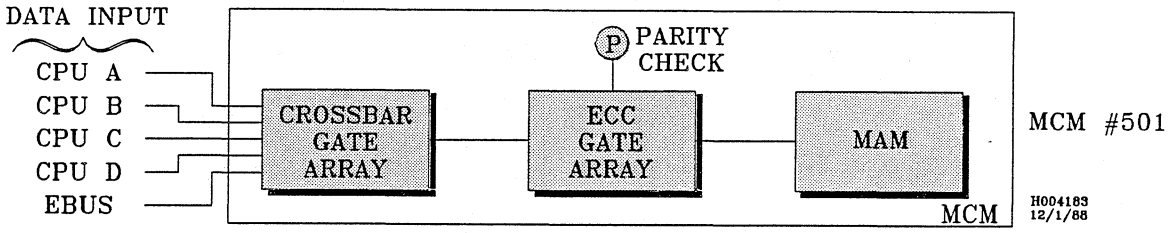
Most likely failed boards (listed alphabetically):

- **MAM**—contains DRAM components
- **MCM**—contains ECC gate arrays, data buses, and MAMs

```
MCMxy/MCM: [#501] ECC unit %d: parity error:
MCMxy/MCM: Write data: %08X, Parity: %02X
```

Description of Error: The MCM has detected a parity error on write data from one of the five ports during a memory request.

Figure 7–30, MCM #501 data flow



Cause and Disposition: Data comes from ASP, PIA, or SP2/4.

If only an MCM #501 error is reported, suspect first the ASP; however, the other boards listed may still be at fault. The *dump_win_q* ISCN function in *mem_func* can be used to determine which port made the request.

If this error is accompanied by a CPX #400 error, suspect PIA or SP2. The MCM only checks parity when it is addressed; if it detects an error, it was addressed. The CPX checks parity constantly and if it is not addressed, it defaults to the EBUS. Therefore, if both a CPX #400 and a MCM #501 error are reported, the data came to the MCM from the EBUS, which is likely at fault.

Run diagnostics *cpu_4030* subtests 200–240 to identify CPU 0, 1, 2, or 3 as the source of bad data, and then *io_4000* to test the EBUS port.

Most likely failed boards (listed alphabetically):

- **ASP**—Possible data source
- **Backplane**—Data paths from ASP, PIA, SP2, CPX
- **CPX**—Can only affect even memory
- **PIA**—Possible data source
- **MCM**—Failed crossbar gate arrays, connectors
- **SP2**—Possible data source

7.2.6 PIA Hard Errors

The following messages are displayed when a hard error is reported by the PIA board.

> **NOTE**
>
> x denotes CPU (A, B, C, or D)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

`PIA: [#600] PBUS arbiter idle/EBUS arbiter expecting write data`

Description of Error: A sequence or state machine error, the PIA is expecting write data when the PBUS is idle. This error is obsolete for PIA boards at or beyond revision level G.

Cause and Disposition: This error is usually caused by a write data parity error (soft error) or by the HSP. Systems in which an HSP is installed should have a PIA at or beyond revision level G.

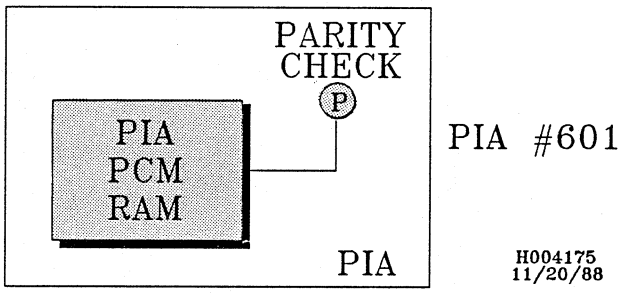
Most likely failed board:

- **PIA**—PIA needs uprev to G or later

```
PIA: [#601] PCM RAM parity error detected. Addr<31..22>: %03x Data: %0x Par:
%0x
```

Description of Error: Parity checker detected error in output of PIA copy of the PCM RAM. Error data and address in PIA log ring.

Figure 7–31, PIA #601 data flow



Cause and Disposition: Most likely causes: PIA copy of PCM RAM not loaded or corrupted, or hardware failure. Run *mminit* and then diagnostic *pia_4000* subtest 350, PCM RAM test.

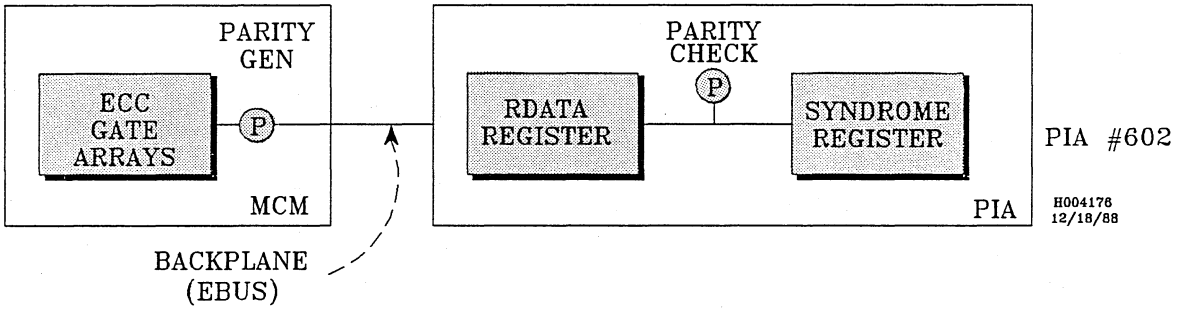
Most likely failed boards:

- **PIA**—Contains PCM RAMS

```
PIA: [#602] EBUS read data parity error
PIA: read data: %08x %08x read parity: %02x Syndrome: %02x
```

Description of Error: This message indicates an EBUS parity error in the data from memory (MCM error). The **%08x** entry indicates the data and **%02** indicates which byte suffered the error by showing which parity bit is **ON**. It is not possible to identify which MCM (odd or even) was the source of the data from the parity bit.

Figure 7–32, PIA #602 data flow



Cause and Disposition: The most likely cause of this error is a PIA that is not initialized (run *sysreset*). There could also be bad data from the MCM, incorrect bus timing, or a PIA hardware failure. Run diagnostics *pia_4000* subtest 101 to check PIA functions.

Most likely failed boards (listed alphabetically):

- **Backplane**—Contains EBUS
- **CPX**—Drives EBUS
- **MCM**—Source of bad data
- **PIA**—Board on which data parity is checked

`PIA: [#603] Interrupt arbitor error`

Description of Error: Interrupt arbiter sequence error.

Cause and Disposition: This error can be caused when an interrupt acknowledge is present while no interrupt is active, or by the deassertion of an interrupt request during processing of the interrupt. Run diagnostics *pia_4000* subtest 550 and *io_4000* subtest 200.

Most likely failed boards (listed alphabetically):

- **ASP**
- **Backplane**
- **CCU**
- **CPX**
- **PIA**

7.2.7 VPC Hard Errors

The following messages are displayed when a hard error is reported by the VPC board.

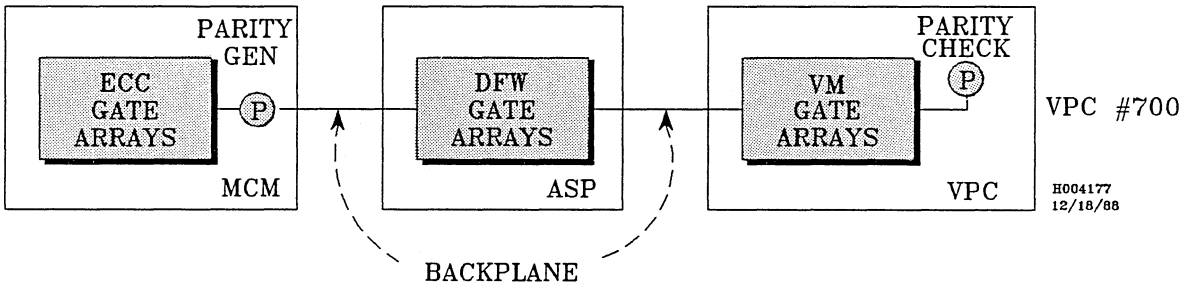
> **NOTE**
>
> x denotes CPU (A, B, C, or D)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

Error #700 is, in fact, a hard error. The remaining errors are all soft errors in the VPC scan ring:

`VPCx/VPC: [#700] VM gate array parity error. Syndrome: %01x`

Description of Error: Parity checker detected error in input staging register of VM gate array.

Figure 7–33, VPC #700 data flow



Cause and Disposition: This error is most likely caused by bad data from the MCM, the ASP DFW gate arrays, or a problem with the VPC VM gate arrays. There could also be a problem with the backplane bus lines. Run diagnostics *cpu_4041* subtests 200–216 and subtests 550–566 to test vector load operations.

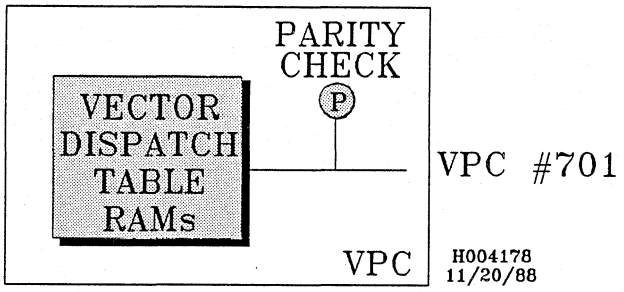
Most likely failed boards (listed alphabetically):

- **Backplane**—Contains data paths
- **ASP**—Contains DFW gate arrays
- **MCM**—Source of data
- **VPC**—Contains VM gate arrays

`VPCx/VPC: [#701] Parity error: Vector dispatch table`

Description of Error: Parity checker detected error in vector dispatch table RAMs.

Figure 7–34, VPC #701 data flow



Cause and Disposition: This error is most likely caused by problems in the vector table dispatch RAMs (failed RAM, slow RAM, or a VPC board defect). Data for RAM is scanned in. Run control store loader with the verify option (**cs -v**).

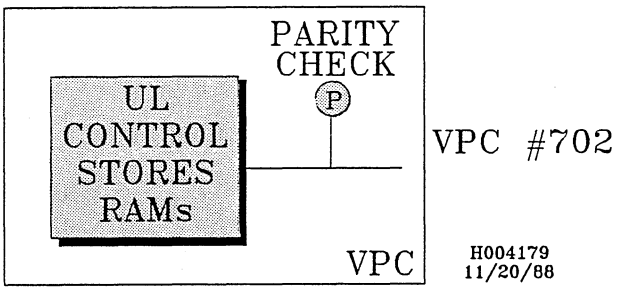
Most likely failed board:

- **VPC**—Contains vector dispatch table RAMs.

`VPCx/VPC: [#702] Parity error: Load/Store control store`

Description of Error: Parity checker detected error in load/store control store RAM data.

Figure 7–35, VPC #702 data flow



Cause and Disposition: This error is most likely caused by problems in the load/store control store RAMs (failed RAM, slow RAM, illegal location read, or a VPC board defect). Data for RAM is scanned in. Run control store loader with the verify option (**cs -v**).

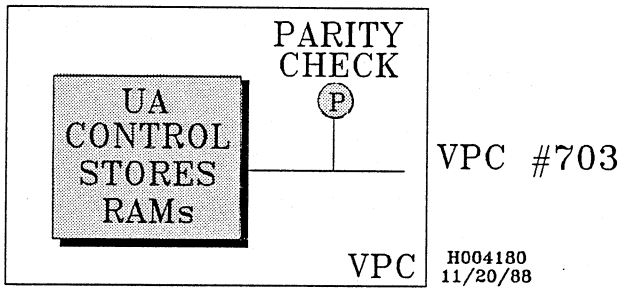
Most likely failed board:

- **VPC**—Contains control store RAMs

`VPCx/VPC: [#703] Parity error: Add control store`

Description of Error: Parity checker detected error in add control store RAMs.

Figure 7–36, VPC #703 data flow



Cause and Disposition: This error is most likely caused by problems in the add control store RAMs (failed RAM, slow RAM, illegal read, or a VPC board defect). Data for RAM is scanned in. Run control store loader with the verify option (**cs -v**).

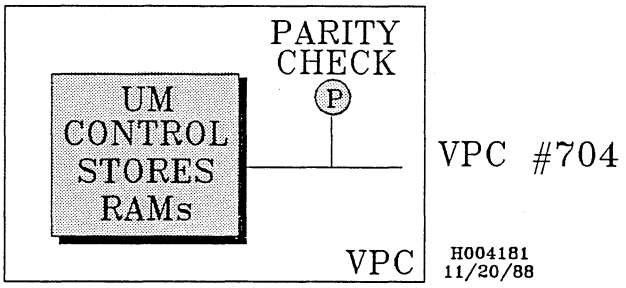
Most likely failed boards (listed alphabetically):

- **VPC**—Contains control store RAMs

`VPCx/VPC: [#704] Parity error: Multiply control store`

Description of Error: Parity checker detected error in multiply control store RAMs.

Figure 7–37, VPC #704 data flow



Cause and Disposition: This error is most likely caused by problems in the multiply control store RAMs (failed RAM, slow RAM, illegal location read, or a VPC board defect). Data for RAM is scanned in. Run control store loader with the verify option (`cs -v`).

Most likely failed board:

- **VPC**—Contains multiply control store RAMs

7.2.8 VPD Hard Errors

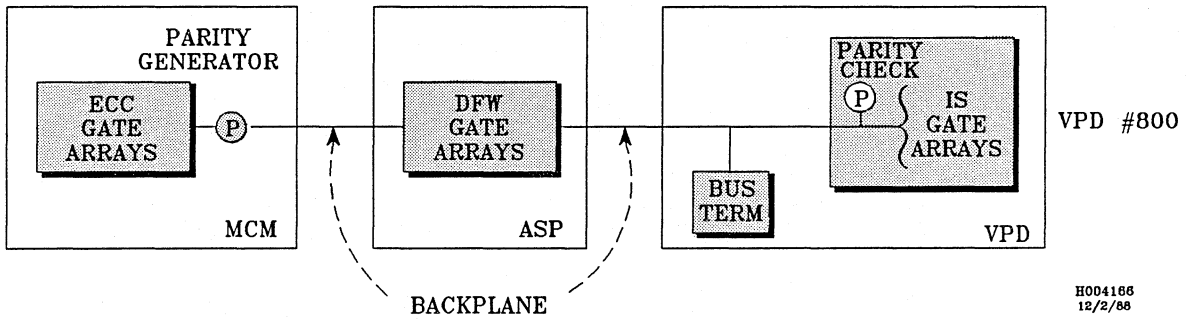
The following messages are displayed when a hard error is reported by the VPD board.

> **NOTE**
>
> x denotes CPU (A, B, C, or D)
>
> If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

```
VPDx/VPD: [#800] is_par_err = %1x
VPDx/VPD: is_stage_data    %08x %08x  %02x *** parity error (%2x) - if error
VPDx/VPD: is_scalar_data   %08x %08x  %02x *** parity error (%2x) - if error
VPDx/VPD: is_even_stage_data %08x %08x
VPDx/VPD: is_odd_stage_data  %08x %08x
VPDx/VPD: is_odd_data      %08x %08x
```

Description of Error: Parity checker detected error in Input Staging (IS) data from main memory via the ASP from the ASP register file or the DCACHE.

Figure 7–38, VPD #800 data flow



Cause and Disposition: Problem could be in the source data (sent from ASP with bad parity), in the backplane buses or ASP or VPD connectors, in the IS gate arrays (on the VPD) which may have failed (bad bits or crosstalk), or failure of the AS.BP.PAR or AS.BP.DATA bus terminations on the VPD. Run diagnostics *cpu_4041* subtests 200–216 and 550–556 to check VPD function.

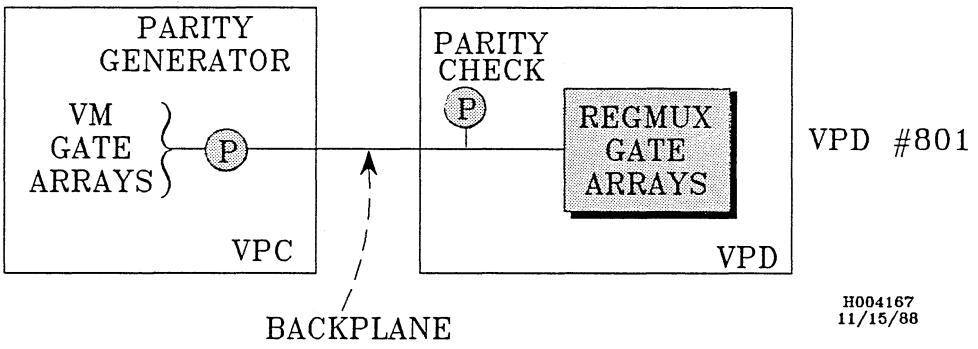
Most likely failed boards (listed alphabetically):

- **ASP**—DFW gate arrays, parity generator
- **Backplane**—Buses and connections to ASP and VPD
- **VPD**—IS gate arrays, AS.BP.PAR and AS.BP.DATA bus terminations

```
VPDx/VPD: [#801] vm_par_err = %1x
VPDx/VPD: os[].cr       %08x %08x
```

Description of Error: Parity checker detected error in Vector Merge (VM) data input to REGMUX (output staging gate arrays) on VPD.

Figure 7–39, VPD #801 data flow



Cause and Disposition: This could be a problem in the VM gate arrays or their parity generators on the VPC, in the VC-VD.data or VC-VD.parity backplane buses, VPC or VPD connectors, or in the REGMUX gate arrays (on the VPD) which may have failed (bad bits or crosstalk). Run diagnostics *cpu_4041* subtests 60–85 and 105–191 to check VPD function.

Most likely failed boards (listed alphabetically):

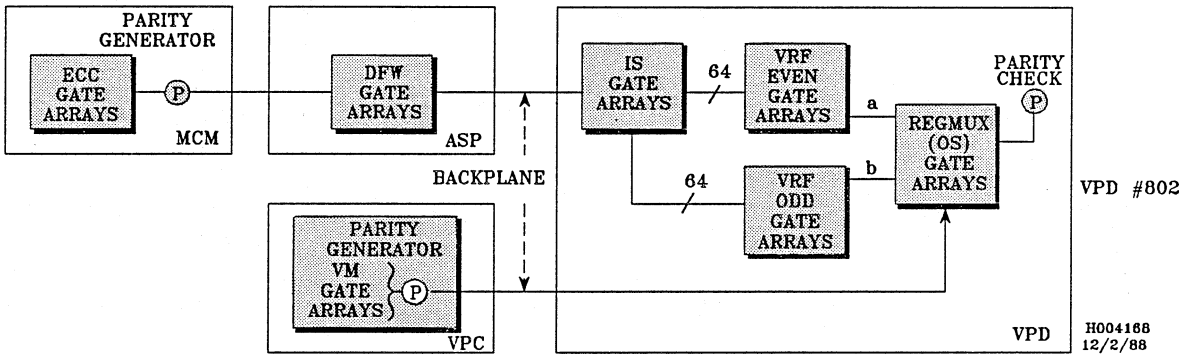
- **Backplane**—Buses and connections to VPC and VPD
- **VPC**—VM gate arrays, parity generator
- **VPD**—REGMUX gate arrays, parity checker

```
VPDx/VPD: [#802] os_par_err = %1x
VPDx/VPD: os_ar   %08x %08x  %02x *** parity error (%2x) - if error - VRF even or IS
VPDx/VPD: os_br   %08x %08x  %02x *** parity error (%2x) - if error - VRF odd or IS
```

Then it checks to see if additional information should be printed:
If there was a os_ar parity error, no os_br parity error, and no vre parity error, dump info for a vre parity error.
If there was a os_br parity error, no os_ar parity error, and no vro parity error, dump info for a vro parity error.
If there was a os_br parity error, a os_ar parity error, and no vre or vro parity error, dump info for a vre and vro parity error.

Description of Error: Parity checker detected error in Output Staging (OS) data.

Figure 7–40, VPD #802 data flow



Cause and Disposition: This could be a problem in the IS gate arrays, the VRF gate arrays, or the REGMUX gate arrays (bad bits or crosstalk). Run diagnostics *cpu_4041* subtests 200–216, 330–416, and 430–486 to check VPD function.

Most likely failed board:

- **VPD**—Contains IS, VRF, and REGMUX gate arrays

```
VPDx/VPD: [#803] vre_par_err = %1x
VPDx/VPDx/VPD: [#804] vro_par_err = %1x
VPDx/VPD: vr†_aop0_     %08x %08x  %02x *** parity error (%2x) - if error
VPDx/VPD: vr†_aop1_     %08x %08x  %02x *** parity error (%2x) - if error
VPDx/VPD: vr†_mop0_     %08x %08x  %02x *** parity error (%2x) - if error
VPDx/VPD: vr†_mop1_     %08x %08x  %02x *** parity error (%2x) - if error
VPDx/VPD: vr†_sop_      %08x %08x  %02x *** parity error (%2x) - if error
```

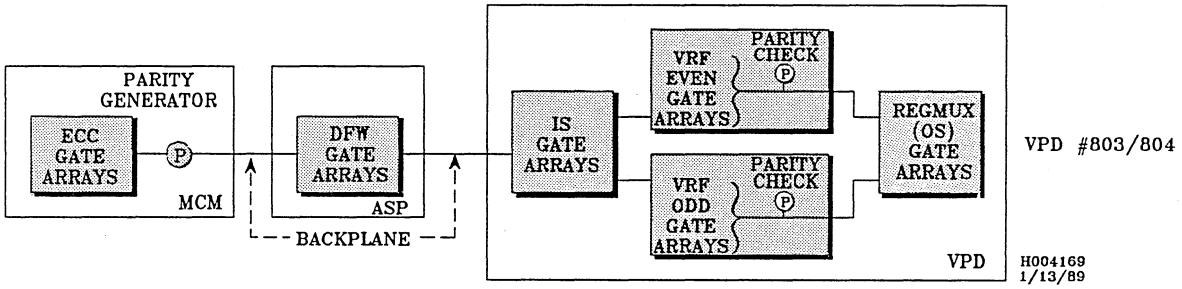
NOTE

† is replaced by **e** for even or **o** for odd.

If simultaneous hard errors are reported, evaluate the problem on the basis of *all* reported errors.

Description of Error: Parity checker detected error in the odd or even Vector Register (VRO or VRE) file.

Figure 7–41, VPD #803/#804 data flow



Cause and Disposition: This could be caused by problems in the IS gate arrays or (more likely) in the VRF odd or even gate arrays, as indicated. Run diagnostics *cpu_4041* subtest 1001 to check VPD function, then dump vector registers with parity (**vp-scn**), and look for errors.

Most likely failed board:

- **VPD**—Contains both the IS gate arrays and the VRF odd and even gate arrays.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 8
I/O Error Messages

8.1 Overview

This chapter lists the various I/O error messages two ways. First are tables containing the error messages and keys to locate the source of the message, and then the following sections contain the error messages arranged by source.

I/O error messages are sent to the service processor errlog file and console. Each is preceded by a preamble indicating the general area of its source and the time the error occurred. Examples:

`SPU_@16:07:51` [*error message text string*]

`CCU`*X*`@_14:56:40` [*error message text string*]

`CPU`*X*`@_02:18:55` [*error message text string*]

where *X* indicates the particular unit sending the message (CPU0–CPU3, CCU0–CCU*N*, etc.).

In addition, it is important to note that only some of the messages listed here appear in *errlog* exactly as shown. These messages consist entirely of text. Example:

```
Controller disabled until reboot.
```

Other messages consist of text and variables, with the variables often appearing as print formats:

```
Multibus cable %d error detected
```

where the `%d` indicates that a decimal character identifying the cable will appear in that location.

Some messages consist almost entirely of print formats:

```
ta%d: cmd: %s esb: %3x %s fsb: %3x %s mux3: %3x %s
```

These messages can be more difficult to locate in the listing since a large portion of the message will consist of variable information. This is especially true when a print format or other variable appears at the beginning of a message.

Some of the more common print formats are listed in the following table:

Table 8–1, Common Print Formats

Format	Result
%c	Single ASCII character
%d	Decimal number
%o	Octal number
%s	Text string
%x	Hexadecimal character

When the % is preceded by a number, the number specifies how many characters will be displayed. For example, `%5x` indicates that five hex characters will appear in the field.

8.2 I/O Error Message List

The following tables list I/O error messages and the keys required to determine the source for each message:

1. Locate the desired message in Table 8–3, I/O Message List (the list is sorted alphabetically)
2. Find its source category number
3. Consult Table 8–2, I/O Error Message Source Key, to determine the source of the message

> **NOTE**
>
> Once the message category number is determined from Table 8–3, I/O Message List, the corresponding discussion in Section 8.3, *I/O Error Messages*, can be located directly.
>
> **Example:**
>
> `Versatec unit %d offline` is a category **22** message. The discussion of this message category is found in Section 8.3.**22**, *IOP Side of Multibus Plotter Driver.*

If the message cannot be found in Table 8–3, I/O Message List, the message may not be from the I/O system (check the messages from other areas), or may as yet be undocumented (please notify Hardware Documentation; refer to the Preface in this volume).

Table 8–2, I/O Error Message Source Key

Category	Source of Message
1	CPU side of Ethernet driver (MIOP/VIOP—not COVUENET) during boot
2	CPU side of Ethernet driver (MIOP/VIOP—not COVUENET) any time
3	VIOP EGOS internal software error
4	CPU side of the UD (User Device Driver)
5	HE driver on the HSP
6	CPU side of Multibus TTY driver
7	CPU side of Console driver
8	CPU side of Multibus disk driver
9	CPU side of Multibus raw Hyperchannel driver
10	CPU side of Multibus Internet Hyperchannel driver
11	CPU side of Multibus DR11W driver
12	CPU side of driver support routines
13	CPU side of Multibus printer driver
14	CPU side of Multibus plotter driver
15	CPU side of Multibus tape driver
16	IOP side (CCU side) of Multibus TTY driver
17	IOP side (CCU side) of Multibus (Xylogics) disk driver
18	IOP side (CCU side) of Ethernet driver
19	IOP side (CCU side) of raw Hyperchannel driver
20	IOP side (CCU side) of Hyperchannel Internet driver
21	IOP side (CCU side) of Multibus printer driver
22	IOP side (CCU side) of Multibus plotter driver
23	IOP side (CCU side) of Multibus tape driver
24	Special IOP side (CCU side) of EGOS *proc_dev* command
25	IOP side support routines for disk and tape large transfer drivers
26	CPU side of VIOP Interphase disk driver (SMD or ESDI)
27	CPU side of VME tape (VTAPE) driver
28	VIOP side of SMD and ESDI disk driver
29	VIOP side of VME tape driver
30	VIOP side of UD (User Device Driver) for VMEbus
31	VME Ethernet driver
32	VIOP side support routines for disk and tape large transfer drivers
33	VIOP EGOS
34	CCU (MIOP, VIOP, HSP) EGOS
35	HSP EGOS
36	Multibus (MIOP) EGOS
37	Multibus (MIOP) EGOS
38	IOP internal
39	VIOP internal

Table 8–3, I/O Error Message List

Error Message	Category
%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE1=%x	25
%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE1=%x	32
%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE2=%x	25
%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE2=%x	32
%s%d error log - cable reset	39
%s: %s:%s [chs=%d/%d/%d]	28
%s: %s:%s [cyl=%d hd=%d sect=%d cnt=%d]	17
%s: can't read bad track table	17
%s: daio unknown state	17
%s: rmap ovflo, lost [%x,%x]	34
%s: submit queue error (dev=%d)	28
%s: unknown reset state %d (dev=%d)	28
%s: unrecoverable disk error	17
%s_mode: DMA not disabled!	35
%s_mode: compliance level != HIA	35
%s_mode: invalid channel chain	35
%s_mode: invalid channel=%d	35
%s_mode: invalid parameter siz=%d	35
%s_mode: zero length chain	35
- Slot %d, Src=%s, Flg:0x%b	38
0x%3", Cache_dbits[flags]	38
68000 Spurious interrupt. Current count = %d	34
68020 Spurious interrupt. Current count = %d	33
ACM-001 Cmd Error %x CSR %x port %x	16
ACM-001 Recursive command error!	16
ACM-001 at %x hung or crashed. Resetting controller!	16
ACM-001 at CSR %x appears to be broken.	16
ACM-001 at CSR %x hung. Status %x PC %x	16
ACM-001 at CSR %x reset by user	16
ACM-001 at CSR %x revived	16
ACM-001 recovery attempt failed.	16
ACM-001 recovery disabled. Reboot required to reset.	16
ADDR ERROR - ACCESS ADDR=0x%x SSW=0x%4	33
Address Save Reg 0x%x, Effective address 0x%x	38
Address=0x%5, Access=%s, Source=%s	38
Buf %d dirty: (i++)	38
Buf %d tag: 0x%b offset=0x%3 (raw_tag=0x%3)	38
Bus error after reset - error logging aborted	36
Bus error after reset - error logging aborted	38
CCU kernel aborted, returning to EPROM code	33

CCU kernel aborted, returning to EPROM code	34
CCU%d panic: surprise breakpoint	34
Cache Controller Error (0x%x, 0x%x): PC=0x%x, SR=0x%4	38
Can't allocate enough IOP windows.	16
Controller disabled until reboot.	16
Controller will be unusable until reboot.	16
Defective trailer packet received on ex%d from %2d:%2d:%2d:%2d:%2d:%2d	2
Device %d interrupt. Interrupt disabled	36
Device %d interrupt. Interrupt disabled	39
Error(s): 0x%b	38
HIA: RegErr=%b	35
HSP 68000 bus error reading CONFIG register	35
HSP Local Memory Parity Error. PC=0x%x, SR=0x%x, ADDR=0x%x	35
HSP: Address=0x%x BusErrLog=0x%b	35
HSP: LstRsp=0x%2, InBusPar=0x%2	35
HSP: PBUS Acc=%s, ErrLog=0x%b	35
IO shared memory MBS failed	12
IO shared memory allocation failed	12
IO shared memory invalid request	12
IO shared memory iop %d failed	12
IOP 68000 Cache Parity Error (0x%x), PC=0x%6, SR=0x%4	36
IOP 68000 Memory Parity Error (0x%x). PC=0x%x, SR=0x%x	36
IOP 68000 Multibus Parity Error (0x%x), PC=0x%6, SR=0x%4	36
Idle process p_page alloc failed	33
Invalid stack frame code = 0x%1 (code_vector=0x%4)	33
Kerntrap: vector %d (0x%2) PC=0x%x SR=0x%4	33
MSG1, name, cmd, err	28
MSG1X, name, cmd, iopb -> i_error, MSGEXCP	28
MSG2, dev, cmd, err	28
MSG2X, cmd, dev, iopb -> i_error, MSGEXCP	28
MSGEND, iopb -> u.phys.i_cyl, iopb -> u.phys.i_hd	28
Multibus cable %d error detected	36
NX Firmware Version %c.%c, Hardware Rev. %c.%c	1
Non-existant VME chassis/cable error	39
PBUS Controller Error (0x%8): %s %s, PC=0x%x, SR=0x%4	38
Please ask CONVEX Field Service to run dev4300.	16
RESET_EXOS error	2
Recovery attempt aborted.	16
Unknown request %d from processor %d	36
VIOP Cache ParErr (0x%x), ADDR=0x%x PC=0x%6, SR=0x%4	39
VIOP Local Mem ParErr (0x%x). ADDR=0x%x PC=0x%x, SR=0x%x	39
VIOP VME ParErr (0x%x), ADDR=0x%x PC=0x%6, SR=0x%4	39
VME cable %d error ", cable)	39
VME cable %d log ((reqid >> 3) & 1)	39
Versatec unit %d not ready	22

Versatec unit %d offline	22
Versatec unit %d out of paper	22
Window 0x%x, Map register=0x%x (0x%x)	38
Wired memory map full	12
Wired memory map full	13
alloc_drvr: channel %d already allocated to device %d	35
alloc_drvr: no more devices available	35
bad rmfree	34
bp %x, pte1 %x, cnt %d	25
bp %x, pte2 %x, cnt %d	25
btinit: bad block table > data window area	17
btinit: unknown state %d	17
btio: unknown state %d	17
ca log buffer wrap	16
ca_dev_service: invalid response; resetting ctlr.	16
ca_free_msg: message return error	16
ca_reset_recovery: Can't get tty lock!	16
caattach: Attach failed for %d/%d/0x%x %d	6
caattach: Attach failed. error=%d	6
caattach: ignored attempt to redefine unit %d	16
caattach: illegal ACM-001 unit %d type %s	16
caattach: last probe failed, attach ABORT	6
cacontrol: Illegal request code %d	16
cacontrol: tp == NULL	16
caprobe: Configure failed. error=%d	6
caprobe: database size %d > limit of 2048	16
case MBS_ERROR: ud driver: can't return message	30
catimeout: ACM-001 at CSR %x appears hung.	16
chnl_eprnt: HSP err=0x%b	35
chnl_eprnt: HSP err=0x%b	35
chnl_eprnt: USR err1=0x%4, err2=0x%4	35
chnl_error: HIA err1=0x%b	35
chnl_error: HIA err2=0x%b	35
chnl_error: HSP err=0x%b	35
cnctl: message free error %x	7
cnctl: tm_send failure %x	7
console_init: message free error %x	7
console_init: tm_send failure %x	7
controller %s not configured	11
controller %s not configured	13
controller %s not configured	15
controller %s not configured	9
couldn't enqueue NET_ADDRS request	31
couldn't enqueue NET_STSTCS request	31
create_process: Device %d, Illegal call	33

create_process: Invalid priority %d	33
create_process: Proc table full!	33
create_process: p_page allocate failed!	33
da%d: No space for bad block table	17
da: double error (cntlr_dev=%d)	17
daattach: allocate of DEV_BSIZE memory failed	17
daattach: allocate of IOPB memory failed	17
daattach: command memory allocate failed	17
daattach: pq_fm_alloc error=%d	8
daattach: pq_fm_free error=%d	8
daclose: pq_fm_alloc error=%d	8
daclose: pq_fm_free error=%d	8
dacmd: dp -> b_done	17
dacmd: message error %d	17
dacmd: return message error	17
dacmd: rtn_msg error in strategy	17
dacmd: unknown command code %d	17
dacmd: unknown state %d	17
dacontrol: CONF message return failed	17
dacontrol: Unknown control case %d	17
dacontrol: message return error	17
dahwintr: dev=%d, active_cmds=%d, reset	17
dahwintr: dev=%d, busy change, times=%d	17
dahwintr: dev=%d, none active, times=%d (%d)	17
daintr: pq_fm_free error=%d	8
daio:	17
daioctl: pq_fm_alloc error=%d	8
daioctl: pq_fm_free error=%d	8
damain: dev=%d, add_request < 0!	17
daopen: pq_fm_alloc error=%d	8
daopen: pq_fm_free error=%d	8
daprobe: MIOP configure failed. error=%d	8
daprobe: database size %d > limit of 2048	17
daprobe: pq_fm_free error=%d	8
daprobe: sysgen limit NDA exceeded for cntlr %d/%d/0x%x	8
dapull: warning - buf not enqueued	8
dastart: dev=%d - (cl != 1) (cl=%d)	17
dastart: dev=%d, attn req ack set (stat=%d)	17
dastart: start at 0x%x	17
dastrategy: Unknown state %d	17
dastrategy: pq_fm_alloc error=%d	8
datimeout: dev=%d stat=%2	17
dd: Bus error detected by cntlr (cntlr_dev=%d)	28
dd: Controller bus error (dev=%d)	28
dd_reset_cntlr: Dev %d busy before reset (%4)	28

ddattach: allocate of DATA memory failed	28
ddattach: allocate of IOPB memory failed	28
ddattach: pq_fm_alloc error=%d	26
ddattach: pq_fm_free error=%d	26
ddclose: pq_fm_alloc error=%d	26
ddclose: pq_fm_free error=%d	26
ddcontrol: dq_recv error	28
ddcontrol: return message error	28
ddhwproc: empty submit queue entry @ %d	28
ddintr: pq_fm_free error=%d	26
ddio: unexpected status (0x%x) in IOPB	28
ddiocmd(%d): dq_recv error	28
ddiocmd: command %d invalid	28
ddiocmd: return message error	28
ddioctl: pq_fm_alloc error=%d	26
ddioctl: pq_fm_free error=%d	26
ddiopberr: No IOPB error detected (e=0x%x)	28
ddmain: database size %d > limit of 2048	28
ddmain: ddcmd (dev, db, 0) returned!	28
ddmain: ddcmd (dev, db, 1) returned!	28
ddmain: ddhwproc returned!	28
ddopen: pq_fm_alloc error=%d	26
ddopen: pq_fm_free error=%d	26
ddprobe: IOP configure failed. error=%d	26
ddprobe: command memory allocate failed	28
ddprobe: pq_fm_alloc error=%d	26
ddprobe: pq_fm_free error=%d	26
ddprobe: sysgen limit NDD exceeded for cntlr %d/%d/0x%x	26
ddpull: warning - buf not enqueued	26
ddstrat: csp -> bp -> b_done	28
ddstrategy: pq_fm_alloc error=%d	26
default: pb: unknown command %d	22
dev_util: Unknown option %d	35
dev_util: Unknown option %d	36
dev_util: Unknown option %d	38
device %s not configured	10
device %s not configured	11
device %s not configured	13
device %s not configured	15
device %s not configured	9
dmacntl.c: dmainvalid: DMA control invalid for compliance level	35
echo_cmd: invalid size (%d)	5
echo_start: ECHO error 0x%b	5
enqueue: message argument is zero.	29
enqueue: nextexec %d, nextpend %d, active %d, execcnt %d	29

ex%d: RESET failed	2
ex%d: address = dd:dd:dd:dd:dd:dd	2
ex_ctlcmd: CNTL pq_fm_alloc error=%d	2
ex_ctlcmd: CNTL pq_fm_free error=%d	2
ex_ctlcmd: CNTL tm_send error=%d	2
ex_postrecvbufs: RECEIVE pq_fm_alloc() error=%d	2
ex_postrecvbufs: RECV msg free error=%d	2
ex_postrecvbufs: RECV tm_send error=%d	2
ex_postrecvbufs: not posting buffer %d	2
exalloc: common use window allocate failed	18
exalloc: debugging page allocate failed	18
exalloc: exconfig failed with status of %d	18
exalloc: space allocation for configuration msg failed	18
exalloc: space allocation for queue headers failed	18
exalloc: space allocation for reply queue failed	18
exalloc: space allocation for request queue failed	18
exattach: IOCNTL tm_send error=%d	1
exattach: controller configure failed, attach aborted	1
exattach: error allocating message	1
exattach: error freeing message	1
excint: CNTL pq_fm_free error=%d	2
excint: GET_ETHER_STATS failed with status of %d	2
exconfig: EXOS configuration error %2x	18
exconfig: timeout waiting for configuration completion	18
excontrol(): MBS_ERROR flushing RECEIVE_SUBQUE	18
excontrol(): message return error	18
excontrol(): message return failed	18
excontrol(): rtn_msg failed flushing RECEIVE_SUBQUE	18
excontrol: INITIALIZE cmd, exsetmode failed	18
exgetstat: cannot get window for request	18
exhwintr: ADDR_PUT_STATE error. state=%d	18
exhwintr: ADDR_PUT_STATE exsend() error	18
exhwintr: BR3: bad brs_m.state %d	18
exhwintr: GET_STAT_STATE NET_ADDRS error %2	18
exhwintr: GET_STAT_STATE error. state=%d	18
exhwintr: GET_STAT_STATE exsend() error	18
exhwintr: NET_ADDRS error %2	18
exhwintr: NET_MODE error %2	18
exhwintr: NET_RECV error %2	18
exhwintr: NET_STSTCS error %2	18
exhwintr: RECEIVE error %2	18
exhwintr: RECEIVE reply with no valid buffer	18
exhwintr: RS1: NET_MODE error %2	18
exhwintr: RS1: exsend() erro	18
exhwintr: RS2: NET_RECV error %2	18

exhwintr: TRANSMIT reply with no valid buffer	18
exhwintr: XMIT_STATE message return error	18
exhwintr: XMIT_STATE message return failed	18
exhwintr: could not auto reset controller	18
exhwintr: could not reconfig controller	18
exhwintr: illegal state machine %d	18
exhwintr: message return error	18
exhwintr: message return error	18
exhwintr: message return failed	18
exhwintr: message return failed	18
exhwintr: nx200 reply queue message overflow	18
exinit: ex_ctlcmd returned error = %d	2
exinit: message return error	18
exinit: message return failed	18
exioctl: could not initialize ex%	2
exmain: no trace buffers	18
exoutput: ex%d: can't handle af%d	2
exprobe: CCU CONF tm_send failed. error=%d	1
exprobe: PROBE tm_send failed. error=%d	1
exprobe: RECV buffer allocation for %d/%d/0x%x failed	1
exprobe: XMIT buffer allocation for %d/%d/0x%x failed	1
exprobe: could not allocate message buffer	1
exprobe: error returning message	1
exprobe: incorrect conf_msg type	1
exprobe: sysgen limit exceeded for cntlr %d/%d/0x%x	1
exreceive: mbuf pointer is null	18
exreceive: physical address = 0	18
exreceive: window allocation failed	18
exrint: OFFLINE pq_fm_free error = %d	2
exrint: RECV pq_fm_free error=%d	2
exrint: RECV tm_send error=%d	2
exsend: message enqueue timeout	18
exstart: XMIT pq_fm_alloc error=%d	2
exstart: XMIT pq_fm_free error=%d	2
exstart: XMIT tm_send error=%d	2
extint: CNTL pq_fm_free error=%d	2
extint: transmit error 0x%x	2
extransmit(): window allocation failed	18
extransmit: dblks[%d].physaddr = 0	18
extransmit: mbuf pointer or numblk error (%8x - %d)	18
flush_subqueue(): retries, queue = %d	31
flush_subqueue(): rtn_msg(), queue = %d	31
fork: Illegal call	33
fork: P_page allocate failed!	33
fork: Proc table full!	33

fork: dev=%d, p_estack alloc failed	33
free_msg: MBS error	39
free_msg: mbs error	35
free_msg: mbs error	36
free_msg: mbs error	4
gbl_event: compliance level error	35
get_slot_mask: Invalid chassis/slot = %d/%d	32
hchnlalloc: channel %d already allocated to device %d	35
hchnldisable: invalid channel access	35
hchnlenable: invalid channel access	35
hchnlenable: sdr0 == 0	35
hchnleprnt: invalid channel access	35
hchnlerror: invalid channel access	35
hchnlexaddr: invalid channel access	35
hchnlresid: invalid channel access	35
hchnltstintr: invalid channel access	35
he_gblevent: error register = 0x%x	5
heattach: Attach failed for %d/%d	5
heattach: Attach failed. error=%d	5
heattach: Channel configure error=%d	5
heattach: Channel configure failed for %d/%d	5
heattach: last hsp failed, attach ABORT	5
heattach: pq_fm_alloc error=%d	5
heattach: pq_fm_free error=%d	5
hecontrol: error in rtn_msg	5
hehwintr: channel %d interrupt unexpected	5
hehwintr: error in rtn_msg	5
heio: pq_fm_alloc error=%d	5
heio: pq_fm_free error=%d	5
heioctl: pq_fm_alloc error=%d	5
heioctl: pq_fm_free error=%d	5
hemain: error in rtn_msg	5
heprobe: sysgen limit NHE exceeded	5
hiamode: invalid parameter cmd=%d	35
hiasetup: compliance level != HIA	35
hiaxfer%d: compliance level != HIA	35
hiaxfer??: invalid rate %d	35
hsp_dev: Unknown request %d from processor %d	35
hsp_testdrvr.c: tstdrvr: pq_fm_free @0x%x failed	5
hy%d awkward status %x	20
hy%d: can't handle af%d	10
hy: %s illegal parameter %d	10
hy: %s illegal parameter %d	9
hy: %s not configured	10
hy: %s not configured	9

hŷ: Change in Hardware Status	19
hy: controller %s not configured	9
hy: controller %s not present	10
hy: controller %s not present	9
hy: no iobsema available for %s	10
hy: no iobsemas available for %s	10
if(if_hyprintfs)	10
if_hy:controller %s not configured	10
ik: %s illegal parameter %d	11
ik: %s not configured	11
ik: controller %s not configured	11
ik: controller %s not present	11
ik: read not DONE, status %x	20
iop_dev: Unknown request %d from processor %d	39
iop_ta.c(tamain): null buf ** in DONE	22
lmap_w_mb: mapping to invalid page %d	37
lmap_w_mb: mapping to main memory address 0x%x	37
lookup_da_type: unsupported type %s	17
lookup_dd_type: unsupported type %s	28
lv12cenable: compliance level error	35
lv12cenable: invalid channel access	35
lv12cenable: sdr0 == 0	35
lv12cerror: compliance level error	35
lv1_4ctstintr: compliance level error	35
lv1_4ctstintr: invalid channel access	35
lv34cenable: compliance level error	35
lv34cenable: invalid channel access	35
lv34cenable: sdr0 == 0	35
lv34cerror: compliance level error	35
lv34cerror: invalid channel access	35
lv34noroutine: compliance level error	35
lvlsetup: compliance level error (lvl=%d)	35
lvlu_intr_dflt: Unassigned channel %d interrupt, disabled	35
lvluintr: unacknowledged user interrupt (0x%x)	35
map_w_mb: mapping to invalid page %d	37
map_w_mb: mapping to main memory address 0x%x	37
map_w_vb: mapping invalid window %d	39
map_w_vb: mapping to main memory address 0x%x	39
map_wndw: mapping to invalid page %d	35
map_wndw: mapping to main memory address 0x%x	35
mtstr, a1, a2, a3, a4, a5, a6, a7, a8, a9	18
nudprobe: bad configuration msg type: %d	4
out of logical address space	25
pa%d printer not ready	21
pa: %s illegal parameter %d	13

pa: %s not configured	13
pa: controller %s not configured	13
pa: controller %s not present	13
pb_rtn_msg: return message error (%s)	22
pbattach: Attach failed for %d/%d/0x%x %d	14
pbattach: Attach failed. error=%d	14
pbattach: pq_fm_alloc error=%d	14
pbattach: pq_fm_free error=%d	14
pbattach: unit not attached; incorrect type	14
pbattach: unit not attached; no controller	14
pbattach: unit not attached; too many units	14
pbprobe: Controller probe failed. error=%d	14
pbprobe: IOP configure failed. error=%d	14
pbprobe: driver probe error	14
pbprobe: egos probe error = %d	14
pbprobe: pq_fm_alloc error=%d	14
pbprobe: pq_fm_free error=%d	14
pbprobe: sysgen limit NPB exceeded for cntlr %d/%d/0x%x	14
pdd_init: struct p_page_def too large	33
pdd_pstart: Dev %d, number %d, process returned	33
pg_alloc(%d): rtn=%x, end=%x	34
pg_free(0x%x, %d): freeing invalid page %d, (0x%x)	33
pg_free(0x%x, %d): freeing invalid page %d, (0x%x)	34
pg_free: Freeing invalid page 0x%x, size %d	33
pg_free: Freeing invalid page 0x%x, size %d	34
pg_setup(): end=%x	34
possible printf() on console if an error occurs.	18
possible printf() on console if an error occurs.	31
pq_fm_alloc error=%d	8
pq_intr: pq_recv reports no message?	35
prb_routine: INVALID PARAMETERS (dir=%d,siz=%d)	34
proc_init: invalid compliance level %d (????)	35
proc_init: invalid config value %d	35
pswtch: bad proc pointer in run queue at 0x%x	3
recv: couldn't enqueue RECEIVE request	31
return_message: rtn_msg() error, mbs_buf = %x	31
rmalloc: size <=0	34
s%d error log - cable reset	36
send_intr (isr=0x%8) PENDING REQUEST CLEARED	33
send_intr (isr=0x%8) PENDING REQUEST CLEARED	34
system console by a printf() in exconfig().	18
system console by a printf() in veconfig	31
ta%d: cmd: %s csb: 0x%b esb: 0x%b fsb: 0x%b rtsb: 0x%b	29
ta%d: cmd: %s esb: %3x %s fsb: %3x %s mux3: %3x %s	23
ta%d: command %s never issued.	29

ta%d: mux0: 0x%x mux1: 0x%x mux2: 0x%x mux3: 0x%x	29
ta%d: tape controller chassis %d ccu %d slot %d is not responding.	29
ta%d: tmux0 %3x mux1 %3x mux2 %3x mux3 %3x	23
ta: %s illegal parameter %d	15
ta: %s not configured	15
ta: Unable to return message 0x%x.	29
ta: chain overrun	23
ta: controller %s not configured	15
ta: controller %s not present	15
ta: no memory for the trace buffer, can't reclaim.	29
ta: no memory for the trace buffer, can't resize.	29
ta: no memory for the trace buffer.	29
ta: not enough local pages available.	29
ta: not enough windows available.	29
ta: pg_alloc() failed.	29
ta: unit %d is not ready.	27
ta: unit %d is not ready.	29
ta: unit %d is offline.	27
ta: unit %d is offline.	29
ta: unit %d no write ring.	27
ta: unknown command.	29
ta_viop_probe(): sizeof struct retn_msg > mbs user data space.	27
tahwproc process is exiting.	29
tamain: database size %d > limit of 2048	29
tamodeset: failed due to no temp buffers (l=%x,w=%x)	23
tamsgproc process is exiting.	29
tamsgproc(): dq_recv() error on subqueue %d.	29
taprobe: bad ta_softc size %d	23
testdrvr: message error %d	5
testdrvr: return error %d	5
timeout table overflow	34
tty overrun: Multibus %d CSR %x Port(s)	16
udcmd: timeout waiting on CCU %d response.	4
udjpintr: m 0x%x d->ud_msg 0x%x	4
udprobe: configuration request failed	4
udprobe: failed PROBE request	4
udprobe: sysgen limit NUD exceeded	4
udreprobe: configuration request failed	4
udreprobe: failed PROBE request	4
udreprobe: internal error! bad msg type: %d	4
unexpected ca recovery state %x	16
unmap_w_mb: unmapping invalid window %d	37
unmap_w_vb: unmapping invalid window %d	39
unmap_wndw: mapping to invalid page %d	35
user dev unknown command %d from %d	24

user_intr_dflt: Unassigned channel %d interrupt, disabled	35
userintr: Global error (0x%x)	35
userintr: unacknowledged user interrupt (0x%x)	35
uta%d: no write ring	15
uta%d: not online	15
ve_int_proc(): nx200 reply queue message overflow	31
ve_ioctl_proc() Invalid ioctl cmd %d	31
ve_ioctl_proc(): NET_ADDRS error %2	31
ve_ioctl_proc(): NET_MODE error %2	31
ve_ioctl_proc(): NET_RECV error %2	31
ve_ioctl_proc(): NET_STSTCS error %2	31
ve_ioctl_proc(): cannot get windows for statistics	31
ve_ioctl_proc(): couldn't enqueue NET_ADDRS request	31
ve_ioctl_proc(): couldn't enqueue NET_RECV request	31
ve_ioctl_proc(): dq_recv() error	31
ve_receive_proc(): dq_recv() error	31
ve_receive_proc(): mbuf pointer is null	31
ve_receive_proc(): physical address = 0	31
ve_receive_proc(): window allocation failed	31
ve_transmit_proc(): CONFIG during RESTART failed	31
ve_transmit_proc(): NET_MODE during RESTART failed	31
ve_transmit_proc(): RESET during RESTART failed	31
ve_transmit_proc(): couldn't enqueue TRANSMIT request	31
ve_transmit_proc(): dblks[%d].physaddr = 0	31
ve_transmit_proc(): dblks[%d].physaddr2 = 0	31
ve_transmit_proc(): dq_recv() error	31
ve_transmit_proc(): mbuf pointer or numblk error. %8x - %d	31
ve_transmit_proc(): window allocation failed	31
vealloc(): receive command window allocate failed	31
vealloc(): transmit command window allocate failed	31
vealloc: debugging page allocate failed	31
vealloc: space allocation for configuration msg failed	31
vealloc: space allocation for queue headers failed	31
vealloc: space allocation for reply queue failed	31
vealloc: space allocation for request queue failed	31
vealloc: veconfig failed with status of %d0, status	31
veconfig: EXOS configuration error %2x	31
veconfig: timeout waiting for configuration completion	31
vemain: database size %d > limit of 2048	31
vemain: no trace buffers	31
vemain: ve_int_proc() returned.	31
vemain: ve_ioctl_proc() returned.	31
vemain: ve_receive_proc() returned.	31
vemain: ve_transmit_proc() returned.	31
vesend: message enqueue timeout	31

vesetmode(): couldn't enqueue NET_MODE request	31
virtmap: out of logical address space	25
virtmap: out of logical address space	32
wndw_alloc(%d): rtn=%x	34
wndw_free(%x, %d):	34
wndw_free: attempted to free 0x%x, size %d	33
wndw_free: attempted to free 0x%x, size %d	34

8.3 I/O Error Messages

The following sections contain I/O error messages arranged by source, each with a definition, and cause/disposition information (To Be Determined—TBD).

8.3.1 CPU side of Ethernet Driver — 1

The following messages originate with the CPU side of the Ethernet driver (not the COVUENET driver) for both the MIOP and the VIOP during the boot process.

`NX Firmware Version %c.%c, Hardware Rev. %c.%c`

Description of Error: TBD

Cause and Disposition: TBD

`exattach: IOCNTL tm_send error=%d`

`exattach: controller configure failed, attach aborted`

`exattach: error allocating message`

`exattach: error freeing message`

`exprobe: CCU CONF tm_send failed. error=%d`

`exprobe: PROBE tm_send failed. error=%d`

`exprobe: RECV buffer allocation for %d/%d/0x%x failed`

`exprobe: XMIT buffer allocation for %d/%d/0x%x failed`

`exprobe: could not allocate message buffer`

`exprobe: error returning message`

`exprobe: incorrect conf_msg type`

`exprobe: sysgen limit exceeded for cntlr %d/%d/0x%x`

8.3.2 CPU side of Ethernet Driver — 2

The following messages originate with the CPU side of the Ethernet driver (not the COVUENET driver) for both the MIOP and the VIOP during the boot process or during normal operation.

`Defective trailer packet received on ex%d from %2d:%2d:%2d:%2d:%2d:%2d`

Description of Error: TBD

Cause and Disposition: TBD

`RECV tm_send error=%d`

`RESET_EXOS error`

`ex%d: RESET failed`

`ex%d: address = dd:dd:dd:dd:dd:dd`

`ex_ctlcmd: CNTL pq_fm_alloc error=%d`

`ex_ctlcmd: CNTL pq_fm_free error=%d`

`ex_ctlcmd: CNTL tm_send error=%d`

`ex_postrecvbufs:`

`ex_postrecvbufs: RECEIVE pq_fm_alloc() error=%d`

`ex_postrecvbufs: RECV msg free error=%d`

`ex_postrecvbufs: not posting buffer %d`

`excint: CNTL pq_fm_free error=%d`

`excint: GET_ETHER_STATS failed with status of %d`

`exinit: ex_ctlcmd returned error = %d`

`exioctl: could not initialize ex%`

`exoutput: ex%d: can't handle af%d`

`exrint: OFFLINE pq_fm_free error = %d`

`exrint: RECV pq_fm_free error=%d`

`exrint: RECV tm_send error=%d`

`exstart: XMIT pq_fm_alloc error=%d`

`exstart: XMIT pq_fm_free error=%d`

`exstart: XMIT tm_send error=%d`

`extint: CNTL pq_fm_free error=%d`

`extint: transmit error 0x%x`

8.3.3 VIOP EGOS Messages

The following message originates with the VIOP EGOS and indicate an internal software error.

`pswtch: bad proc pointer in run queue at 0x%x 3p`

Description of Error: TBD

Cause and Disposition: TBD

8.3.4 CPU side of UD

The following messages originate with the CPU side of the UD (User Device Driver), which is part of the optional UWDD (User Written Device Driver) development product.

`free_msg: mbs error`

Description of Error: TBD

Cause and Disposition: TBD

`nudprobe: bad configuration msg type: %d`

`udcmd: timeout waiting on CCU %d response.`

`udjpintr: m 0x%x d->ud_msg 0x%x`

`udprobe: configuration request failed`

`udprobe: failed PROBE request`

`udprobe: sysgen limit NUD exceeded`

`udreprobe: configuration request failed`

`udreprobe: failed PROBE request`

`udreprobe: internal error! bad msg type: %d`

8.3.5 HE Driver on HSP

The following messages originate with the HE (HIA Echo) driver on the HSP. This driver is used internally at CONVEX, primarily for diagnostics and testing of the HIA echo board.

`echo_cmd: invalid size (%d)`

Description of Error: TBD

Cause and Disposition: TBD

`echo_start: ECHO error 0x%b`

`he_gblevent: error register = 0x%x`

`heattach: Attach failed for %d/%d`

`heattach: Attach failed. error=%d`

`heattach: Channel configure error=%d`

`heattach: Channel configure failed for %d/%d`

`heattach: last hsp failed, attach ABORT`

`heattach: pq_fm_alloc error=%d`

`heattach: pq_fm_free error=%d`

`hecontrol: error in rtn_msg`

`hehwintr: channel %d interrupt unexpected`

`hehwintr: error in rtn_msg`

`heio: pq_fm_alloc error=%d`

`heio: pq_fm_free error=%d`

`heioctl: pq_fm_alloc error=%d`

`heioctl: pq_fm_free error=%d`

`hemain: error in rtn_msg`

`heprobe: sysgen limit NHE exceeded`

`testdrvr: message error %d`

`testdrvr: return error %d`

`tstdrvr: pq_fm_free @0x%x failed`

8.3.6 CPU Side of Multibus TTY Driver

The following messages originate with the CPU side of the Multibus TTY driver.

`caattach: Attach failed for %d/%d/0x%x %d`

Description of Error: TBD

Cause and Disposition: TBD

`caattach: Attach failed. error=%d`

`caattach: last probe failed, attach ABORT`

`caprobe: Configure failed. error=%d`

8.3.7 CPU Side of Console Driver

The following messages originate with the CPU side of the console driver.

`cnctl: message free error %x`

Description of Error: TBD

Cause and Disposition: TBD

`cnctl: tm_send failure %x`

`console_init: message free error %x`

`console_init: tm_send failure %x`

8.3.8 CPU Side of Multibus Disk Driver

The following messages originate with the CPU side of the Multibus disk driver.

`daattach: pq_fm_alloc error=%d`

Description of Error: TBD

Cause and Disposition: TBD

`daattach: pq_fm_free error=%d`

`daclose: pq_fm_alloc error=%d`

daclose: pq_fm_free error=%d

daintr: pq_fm_free error=%d

daioctl: pq_fm_alloc error=%d

daioctl: pq_fm_free error=%d

daopen: pq_fm_alloc error=%d

daopen: pq_fm_free error=%d

daprobe: MIOP configure failed. error=%d

daprobe: pq_fm_alloc error=%d

daprobe: pq_fm_free error=%d

daprobe: sysgen limit NDA exceeded for cntlr %d/%d/0x%x

dapull: warning - buf not enqueued

dastrategy: pq_fm_alloc error=%d

8.3.9 CPU Side of Multibus Raw Hyperchannel Driver

The following messages originate with the CPU side of the Multibus raw Hyperchannel driver.

controller %s not configured

Description of Error: TBD

Cause and Disposition: TBD

device %s not configured

hy: %s illegal parameter %d

hy: %s not configured

hy: controller %s not configured

hy: controller %s not present

8.3.10 CPU Side of Multibus Internet Hyperchannel Driver

The following messages originate with the CPU side of the Multibus Internet Hyperchannel

driver.

`%d ", m->m_len`

Description of Error: TBD

Cause and Disposition: TBD

`controller %s not configured`

`device %s not configured`

`hy%d: can't handle af%d`

`hy: %s illegal parameter %d`

`hy: %s not configured`

`hy: %s not configured`

`hy: controller %s not present`

`hy: no iobsema available for %s`

`hy: no iobsemas available for %s`

`if_hy:controller %s not configured`

```
int  if_hyprintfs = 0
```

`m0, "out", if_hyprintfs`

`mprint(m, "in ", if_hyprintfs)`

8.3.11 CPU Side of Multibus DR11W Driver

The following messages originate with the CPU side of the Multibus DR11W driver (Raster Tech interface).

`controller %s not configured`

Description of Error: TBD

Cause and Disposition: TBD

`device %s not configured`

`ik: %s illegal parameter %d`

`ik: %s not configured`

`ik: controller %s not configured`

`ik: controller %s not present`

8.3.12 CPU Side of Driver Support Routines

The following messages originate with the CPU side of driver support routines.

`IO shared memory MBS failed`
Description of Error: TBD

Cause and Disposition: TBD

`IO shared memory allocation failed`

`IO shared memory invalid request`

`IO shared memory iop %d failed`

`Wired memory map full`

8.3.13 CPU Side of Multibus Printer Driver

The following messages originate with the CPU side of the Multibus printer driver.

`Wired memory map full`
Description of Error: TBD

Cause and Disposition: TBD

`controller %s not configured`

`device %s not configured`

`pa: %s illegal parameter %d`

`pa: %s not configured`

`pa: controller %s not configured`

`pa: controller %s not present`

8.3.14 CPU Side of Multibus Plotter Driver

The following messages originate with the CPU side of the Multibus plotter driver.

`pbattach: Attach failed for %d/%d/0x%x %d`

Description of Error: TBD

Cause and Disposition: TBD

`pbattach: Attach failed. error=%d`

`pbattach: pq_fm_alloc error=%d`

`pbattach: pq_fm_free error=%d`

`pbattach: unit not attached; incorrect type`

`pbattach: unit not attached; no controller`

`pbattach: unit not attached; too many units`

`pbprobe: Controller probe failed. error=%d`

`pbprobe: IOP configure failed. error=%d`

`pbprobe: driver probe error`

`pbprobe: egos probe error = %d`

`pbprobe: pq_fm_alloc error=%d`

`pbprobe: pq_fm_free error=%d`

`pbprobe: sysgen limit NPB exceeded for cntlr %d/%d/0x%x`

8.3.15 CPU Side of Multibus Tape Driver

The following messages originate with the CPU side of the Multibus tape driver.

`controller %s not configured`

Description of Error: TBD

Cause and Disposition: TBD

`device %s not configured`

```
ta: %s illegal parameter %d

ta: %s not configured

ta: controller %s not present

ta: controller %s not configured

uta%d: no write ring

uta%d: not online
```

8.3.16 IOP Side of Multibus TTY Driver

The following messages originate with the IOP side (CCU side) of the Multibus TTY driver.

%x

Description of Error: TBD

Cause and Disposition: TBD

```
ACM-001 Cmd Error %x CSR %x port %x

ACM-001 Recursive command error!

ACM-001 at %x hung or crashed. Resetting controller!

ACM-001 at CSR %x appears to be broken.

ACM-001 at CSR %x hung. Status %x PC %x

ACM-001 at CSR %x reset by user

ACM-001 at CSR %x revived

ACM-001 recovery attempt failed.

ACM-001 recovery disabled. Reboot required to reset.

Can't allocate enough IOP windows.

Controller disabled until reboot.

Controller will be unusable until reboot.

Please ask CONVEX Field Service to run dev4300.

Recovery attempt aborted.
```

```
ca log buffer wrap

ca_dev_service: invalid response; resetting ctlr.

ca_free_msg: message return error

ca_reset_recovery: Can't get tty lock!

caattach: ignored attempt to redefine unit %d

caattach: illegal ACM-001 unit %d type %s

cacontrol: Illegal request code %d

cacontrol: tp == NULL

caprobe: database size %d > limit of 2048

catimeout: ACM-001 at CSR %x appears hung.

tty overrun: Multibus %d CSR %x Port(s)

unexpected ca recovery state %x
```

8.3.17 IOP Side of Multibus Disk Driver (Xylogics)

The following messages originate with the IOP side (CCU side) of the Xylogics Multibus TTY driver.

`%s: %s:%s [cyl=%d hd=%d sect=%d cnt=%d]`

Description of Error: TBD

Cause and Disposition: TBD

```
%s: can't read bad track table

%s: daio unknown state

%s: unrecoverable disk error

btinit: bad block table > data window area

btinit: unknown state %d

btio: unknown state %d

da%d: No space for bad block table

da: double error (cntlr_dev=%d)
```

```
daattach: allocate of DEV_BSIZE memory failed

daattach: allocate of IOPB memory failed

daattach: command memory allocate failed

dacmd: dp -> b_done

dacmd: message error %d

dacmd: return message error

dacmd: rtn_msg error in strategy

dacmd: unknown command code %d

dacmd: unknown state %d

dacontrol: CONF message return failed

dacontrol: Unknown control case %d

dacontrol: message return error

dahwintr: dev=%d, active_cmds=%d, reset

dahwintr: dev=%d, busy change, times=%d

dahwintr: dev=%d, none active, times=%d (%d)

damain: dev=%d, add_request < 0!

daprobe: database size %d > limit of 2048

dastart: dev=%d - (cl != 1) (cl=%d)

dastart: dev=%d, attn req ack set (stat=%d)

dastart: start at 0x%x

dastrategy: Unknown state %d

datimeout: dev=%d stat=%2

lookup_da_type: unsupported type %s
```

8.3.18 IOP Side of Ethernet Driver

The following messages originate with the IOP side (CCU side) of the Ethernet driver.

exalloc: common use window allocate failed

Description of Error: TBD

Cause and Disposition: TBD

exalloc: debugging page allocate failed

exalloc: exconfig failed with status of %d

exalloc: space allocation for configuration msg failed

exalloc: space allocation for queue headers failed

exalloc: space allocation for reply queue failed

exalloc: space allocation for request queue failed

exconfig: EXOS configuration error %2x

exconfig: timeout waiting for configuration completion

excontrol(): MBS_ERROR flushing RECEIVE_SUBQUE

excontrol(): message return error

excontrol(): message return failed

excontrol(): rtn_msg failed flushing RECEIVE_SUBQUE

excontrol: INITIALIZE cmd, exsetmode failed

exgetstat: cannot get window for request

exhwintr: ADDR_PUT_STATE error. state=%d

exhwintr: ADDR_PUT_STATE exsend() error

exhwintr: BR3: bad brs_m.state %d

exhwintr: GET_STAT_STATE NET_ADDRS error %2

exhwintr: GET_STAT_STATE error. state=%d

exhwintr: GET_STAT_STATE exsend() error

exhwintr: NET_ADDRS error %2

exhwintr: NET_MODE error %2

exhwintr: NET_RECV error %2

```
exhwintr: NET_STSTCS error %2

exhwintr: RECEIVE error %2

exhwintr: RECEIVE reply with no valid buffer

exhwintr: RS1: NET_MODE error %2

exhwintr: RS1: exsend() erro

exhwintr: RS2: NET_RECV error %2

exhwintr: TRANSMIT reply with no valid buffer

exhwintr: XMIT_STATE message return error

exhwintr: XMIT_STATE message return failed

exhwintr: could not auto reset controller

exhwintr: could not reconfig controller

exhwintr: illegal state machine %d

exhwintr: message return error

exhwintr: message return error

exhwintr: message return failed

exhwintr: message return failed

exhwintr: nx200 reply queue message overflow

exinit: message return error

exinit: message return failed

exmain: no trace buffers

exreceive: mbuf pointer is null

exreceive: physical address = 0

exreceive: window allocation failed

exsend: message enqueue timeout

extransmit(): window allocation failed

extransmit: dblks[%d].physaddr = 0

extransmit: dblks[%d].physaddr = 0
```

`extransmit: mbuf pointer or numblk error (%8x - %d)`

`possible printf() on console if an error occurs.`

`system console by a printf() in exconfig().`

8.3.19 IOP Side of Raw Hyperchannel Driver

The following message originates with the IOP side (CCU side) of the raw Hyperchannel driver.

`iop_hy.c: hy: Change in Hardware Status`

Description of Error: TBD

Cause and Disposition: TBD

8.3.20 IOP Side of Hyperchannel Internet Driver

The following messages originate with the IOP side (CCU side) of the Hyperchannel Internet driver.

`hy%d awkward status %x`

Description of Error: TBD

Cause and Disposition: TBD

`ik: read not DONE, status %x`

8.3.21 IOP Side of Multibus Printer Driver

The following message originates with the IOP side (CCU side) of the Multibus printer driver.

`iop_pa.c: pa%d printer not ready`

Description of Error: TBD

Cause and Disposition: TBD

8.3.22 IOP Side of Multibus Plotter Driver

The following messages originate with the IOP side (CCU side) of the Multibus plotter driver.

`iop_pb.c: Versatec unit %d not ready`

Description of Error: TBD

Cause and Disposition: TBD

`iop_pb.c: Versatec unit %d offline`

`iop_pb.c: Versatec unit %d out of paper`

`iop_pb.c: default: pb: unknown command %d`

`iop_pb.c: pb_rtn_msg: return message error (%s)`

`iop_ta.c: iop_ta.c(tamain): null buf * in DONE`

8.3.23 IOP Side of Multibus Tape Driver

The following messages originate with the IOP side (CCU side) of the Multibus tape driver.

`ta%d: cmd: %s esb: %3x %s fsb: %3x %s mux3: %3x %s`

Description of Error: TBD

Cause and Disposition: TBD

`ta%d:mux0 %3x mux1 %3x mux2 %3x mux3 %3x`

`ta: chain overrun`

`tamodeset: failed due to no temp buffers (l=%x,w=%x)`

`taprobe: bad ta_softc size %d`

8.3.24 Special IOP Side of EGOS *proc_dev* Command

The following is a special message from the IOP side (CCU side) of the EGOS *proc_dev* command.

`iop_user_dev.c: user dev unknown command %d from %d`

Description of Error: TBD

Cause and Disposition: TBD

8.3.25 IOP Side Driver Support Routines

The following messages originate with the IOP side (CCU side) support routines used by disk and tape drivers for large transfers. These routines cause the CCU to do the virtual-to-physical address translation during the transfer process.

`%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE1=%x`

Description of Error: TBD

Cause and Disposition: TBD

`%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE2=%x`

`bp %x, pte1 %x, cnt %d`

`bp %x, pte2 %x, cnt %d`

`out of logical address space`

`virtmap: out of logical address space`

8.3.26 CPU Side of VIOP (Interphase) Disk Driver

The following messages originate with the CPU side of the Interphase VIOP disk driver. The controller may be either SMD or ESDI.

`ddattach: pq_fm_alloc error=%d`

Description of Error: TBD

Cause and Disposition: TBD

`ddattach: pq_fm_free error=%d`

`ddclose: pq_fm_alloc error=%d`

`ddclose: pq_fm_free error=%d`

```
ddintr: pq_fm_free error=%d

ddioctl: pq_fm_alloc error=%d

ddioctl: pq_fm_free error=%d

ddopen: pq_fm_alloc error=%d

ddopen: pq_fm_free error=%d

ddprobe: IOP configure failed. error=%d

ddprobe: pq_fm_alloc error=%d

ddprobe: pq_fm_free error=%d

ddprobe: sysgen limit NDD exceeded for cntlr %d/%d/0x%x

ddpull: warning - buf not enqueued

ddstrategy: pq_fm_alloc error=%d
```

8.3.27 CPU Side of VME (VTAPE) Tape Driver

The following messages originate with the CPU side of the VTAPE tape driver.

```
cpu_viop_ta.c: ta: unit %d is not ready. 4p
```

Description of Error: TBD

Cause and Disposition: TBD

```
cpu_viop_ta.c: ta: unit %d is offline.

cpu_viop_ta.c: ta: unit %d no write ring.

cpu_viop_ta.c: ta_viop_probe(): sizeof struct send_msg > mbs user data
space.
```

8.3.28 VIOP Side of SMD and ESDI Disk Driver

The following messages originate with the VIOP side of the SMD and ESDI disk driver.

`%s: %s:%s [chs=%d/%d/%d]`

Description of Error: TBD

Cause and Disposition: TBD

`%s: submit queue error (dev=%d)`

`%s: unknown reset state %d (dev=%d)`

`MSG1, name, cmd, err`

`MSG1X, name, cmd, iopb -> i_error, MSGEXCP`

`MSG2, dev, cmd, err`

`MSG2X, cmd, dev, iopb -> i_error, MSGEXCP`

`MSGEND, iopb -> u.phys.i_cyl, iopb -> u.phys.i_hd`

`dd: Bus error detected by cntlr (cntlr_dev=%d)`

`dd: Controller bus error (dev=%d)`

`dd_reset_cntlr: Dev %d busy before reset (%4)`

`ddattach: allocate of IOPB memory failed`

`ddcontrol: dq_recv error`

`ddcontrol: return message error`

`ddhwproc: empty submit queue entry @ %d`

`ddio: unexpected status (0x%x) in IOPB`

`ddiocmd(%d): dq_recv error`

`ddiocmd: command %d invalid`

`ddiocmd: return message error`

`ddiopberr: No IOPB error detected (e=0x%x)`

`ddmain: database size %d > limit of 2048`

`ddmain: ddcmd (dev, db, 0) returned!`

`ddmain: ddcmd (dev, db, 1) returned!`

`ddmain: ddhwproc returned!`

```
ddprobe: command memory allocate failed

ddstrat: csp -> bp -> b_done

lookup_dd_type: unsupported type %s

viop_ddattach.c: ddattach: allocate of DATA memory failed
```

8.3.29 VIOP Side of VME Tape Driver

The following messages originate with the VME tape driver.

```
enqueue: message argument is zero.
```

Description of Error: TBD

Cause and Disposition: TBD

```
enqueue: nextexec %d, nextpend %d, active %d, execcnt %d

ta%d: cmd: %s csb: 0x%b esb: 0x%b fsb: 0x%b rtsb: 0x%b

ta%d: command %s never issued.

ta%d: mux0: 0x%x mux1: 0x%x mux2: 0x%x mux3: 0x%x

ta%d: tape controller chassis %d ccu %d slot %d is not responding.

ta: Unable to return message 0x%x.

ta: no memory for the trace buffer, can't reclaim.

ta: no memory for the trace buffer, can't resize.

ta: no memory for the trace buffer.

ta: not enough local pages available.

ta: not enough windows available.

ta: pg_alloc() failed.

ta: unit %d is not ready.

ta: unit %d is offline.

ta: unknown command.

tahwproc process is exiting.
```

`tamain: database size %d > limit of 2048`

`tamsgproc process is exiting.`

`tamsgproc(): dq_recv() error on subqueue %d.`

8.3.30 VIOP Side of UWDD UD Driver

The following message originates with the VIOP side (CCU side) of the UD (User Device Driver), which is part of the optional UWDD (User Written Device Driver) development product.

`viop_ud.c: case MBS_ERROR: ud driver: can't return message 3p`

Description of Error: TBD

Cause and Disposition: TBD

8.3.31 VME Ethernet Driver

The following messages originate with the VME Ethernet driver.

`couldn't enqueue NET_ADDRS request`

Description of Error: TBD

Cause and Disposition: TBD

`couldn't enqueue NET_STSTCS request`

`flush_subqueue(): retries, queue = %d`

`flush_subqueue(): rtn_msg(), queue = %d`

`possible printf() on console if an error occurs.`

`recv: couldn't enqueue RECEIVE request`

`return_message: rtn_msg() error, mbs_buf = %x`

`system console by a printf() in veconfig`

`ve_int_proc(): nx200 reply queue message overflow`

`ve_ioctl_proc() Invalid ioctl cmd %d`

`ve_ioctl_proc(): NET_ADDRS error %2`

```
ve_ioctl_proc(): NET_ADDRS error %2

ve_ioctl_proc(): NET_MODE error %2

ve_ioctl_proc(): NET_RECV error %2

ve_ioctl_proc(): NET_STSTCS error %2

ve_ioctl_proc(): cannot get windows for statistics

ve_ioctl_proc(): couldn't enqueue NET_ADDRS request

ve_ioctl_proc(): couldn't enqueue NET_RECV request

ve_ioctl_proc(): dq_recv() error

ve_receive_proc(): dq_recv() error

ve_receive_proc(): mbuf pointer is null

ve_receive_proc(): physical address = 0

ve_receive_proc(): window allocation failed

ve_transmit_proc(): CONFIG during RESTART failed

ve_transmit_proc(): NET_MODE during RESTART failed

ve_transmit_proc(): RESET during RESTART failed

ve_transmit_proc(): couldn't enqueue TRANSMIT request

ve_transmit_proc(): dblks[%d].physaddr = 0

ve_transmit_proc(): dblks[%d].physaddr2 = 0

ve_transmit_proc(): dq_recv() error

ve_transmit_proc(): mbuf pointer or numblk error.(%8x - %d

ve_transmit_proc(): window allocation failed

vealloc(): receive command window allocate failed

vealloc(): transmit command window allocate failed

vealloc: debugging page allocate failed

vealloc: space allocation for configuration msg failed

vealloc: space allocation for queue headers failed

vealloc: space allocation for reply queue failed
```

`vealloc: space allocation for request queue failed`

`vealloc: veconfig failed with status of %d`

`veconfig: EXOS configuration error %2x`

`veconfig: timeout waiting for configuration completion`

`vemain: database size %d > limit of 2048`

`vemain: no trace buffers`

`vemain: ve_int_proc() returned.`

`vemain: ve_ioctl_proc() returned.`

`vemain: ve_receive_proc() returned.`

`vemain: ve_transmit_proc() returned.`

`vesend: message enqueue timeout`

`vesetmode(): couldn't enqueue NET_MODE request`

8.3.32 VIOP Side Driver Support Routines

The following messages originate with the VIOP side (CCU side) support routines used by disk and tape drivers for large transfers. These routines cause the CCU to do the virtual-to-physical address translation during the transfer process. Note that Ethernet does not use *virtmap*.

`%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE1=%x`

Description of Error: TBD

Cause and Disposition: TBD

`%s virtual=%x, SDR=%x, <SDR[%d]>=%x, PTE2=%x`

`get_slot_mask: Invalid chassis/slot = %d/%d`

`virtmap: out of logical address space`

8.3.33 VIOP EGOS Messages

The following messages originate with the VIOP EGOS. Note that the `%8` is a print format that prints 8 hex digits, e.g., `05fff230`.

```
68020 Spurious interrupt. Current count = %d
```

Description of Error: TBD

Cause and Disposition: TBD

```
ADDR ERROR - ACCESS ADDR=0x%x SSW=0x%4
 <d0-d3> %8 %8 %8 %8
 <d4-d7> %8 %8 %8 %8
 <a0-a3> %8 %8 %8 %8
 <a4-a7> %8 %8 %8 %8

BUS ERROR - ACCESS ADDR=0x%x SSW=0x%4
 <d0-d3> %8 %8 %8 %8
 <d4-d7> %8 %8 %8 %8
 <a0-a3> %8 %8 %8 %8
 <a4-a7> %8 %8 %8 %8

CCU kernel aborted, returning to EPROM code

Invalid stack frame code = 0x%1 (code_vector=0x%4)

Kerntrap: vector %d (0x%2) PC=0x%x SR=0x%4

panic: surprise breakpoint

pdd_createp.c: create_process: Device %d, Illegal call

pdd_createp.c: create_process: Invalid priority %d

pdd_createp.c: create_process: Proc table full!

pdd_createp.c: create_process: p_page allocate failed!

pdd_fork.c: fork: Illegal call

pdd_fork.c: fork: P_page allocate failed!

pdd_fork.c: fork: Proc table full!

pdd_fork.c: fork: dev=%d, p_estack alloc failed

pdd_init.c: pdd_init: Idle process p_page alloc failed

pdd_init.c: pdd_init: struct p_page_def too large

pdd_pstart.c: pdd_pstart: Dev %d, number %d, process returned

pg_free(0x%x, %d): freeing invalid page %d, (0x%x)

pg_free: Freeing invalid page 0x%x, size %d
```

```
send_intr (isr=0x%8) PENDING REQUEST CLEARED

wndw_free: attempted to free 0x%x, size %d
```

8.3.34 CCU EGOS Messages

The following messages originate with a CCU (MIOP, VIOP, HSP, etc.). Although some of these messages may be the result of a driver software problem or a controller hardware problem, they are reported by EGOS. Note that %8 is a print format that prints 8 hex digits, e.g., o5fff230.

```
%s ERROR - ACCESS ADDR=0x%x, FC=0x%4, IR=0x%4
 <d0-d3> %8 %8 %8 %8
 <d4-d7> %8 %8 %8 %8
 <a0-a3> %8 %8 %8 %8
 <a4-a7> %8 %8 %8 %8
```

Description of Error: TBD

Cause and Disposition: TBD

```
%s: rmap ovflo, lost [%x,%x]

68000 Spurious interrupt. Current count = %d

CCU kernel aborted, returning to EPROM code

CCU%d panic: surprise breakpoint

Kerntrap: vector %d (0x%2) PC=0x%x SR=0x%4 USP=0x%x

bad rmfree

pg_alloc(%d): rtn=%x, end=%x

pg_free(0x%x, %d): freeing invalid page %d, (0x%x)

pg_free: Freeing invalid page 0x%x, size %d

pg_setup(): end=%x

prb_routine: INVALID PARAMETERS (dir=%d,siz=%d)

rmalloc: size <=0

send_intr (isr=0x%8) PENDING REQUEST CLEARED

timeout table overflow

wndw_alloc(%d): rtn=%x
```

```
wndw_free(%x, %d):

wndw_free: attempted to free 0x%x, size %d
```

8.3.35 HSP EGOS Messages

The following messages originate with the HSP EGOS.

```
%s_mode: DMA not disabled!
```

Description of Error: TBD

Cause and Disposition: TBD

```
%s_mode: compliance level != HIA

%s_mode: invalid channel chain

%s_mode: invalid channel=%d

%s_mode: invalid parameter siz=%d

%s_mode: zero length chain

HIA: RegErr=%b

HSP 68000 bus error reading CONFIG register

HSP Local Memory Parity Error. PC=0x%x, SR=0x%x, ADDR=0x%x

HSP: Address=0x%x BusErrLog=0x%b

HSP: LstRsp=0x%2, InBusPar=0x%2

HSP: PBUS Acc=%s, ErrLog=0x%b

alloc_drvr: channel %d already allocated to device %d

alloc_drvr: no more devices available

chnl_eprnt: HSP  err=0x%b

chnl_eprnt: HSP  err=0x%b

chnl_eprnt: USR err1=0x%4, err2=0x%4

chnl_error: HIA err1=0x%b

chnl_error: HIA err2=0x%b
```

```
chnl_error: HSP  err=0x%b

dev_util: Unknown option %d

dmainvalid: DMA control invalid for compliance level

free_msg: mbs error

gbl_event: compliance level error

hchnlalloc: channel %d already allocated to device %d

hchnldisable: invalid channel access

hchnlenable: invalid channel access

hchnlenable: sdr0 == 0

hchnleprnt: invalid channel access

hchnlerror: invalid channel access

hchnlexaddr: invalid channel access

hchnlresid: invalid channel access

hchnltstintr: invalid channel access

hiamode: invalid parameter cmd=%d

hiaxfer%d: compliance level != HIA

hiaxfer??: invalid rate %d

hsp_dev: Unknown request %d from processor %d

if (ccu_debuglvl >= lvl)

lv12cenable: compliance level error

lv12cenable: invalid channel access

lv12cenable: sdr0 == 0

lv12cerror: compliance level error

lv12cerror: invalid channel access

lv1_4ctstintr: compliance level error

lv1_4ctstintr: invalid channel access

lv34cenable: compliance level error
```

```
lv34cenable: invalid channel access

lv34cenable: sdr0 == 0

lv34cerror: compliance level error

lv34cerror: invalid channel access

lv34noroutine: compliance level error

lvlsetup: compliance level error (lvl=%d)

lvlu_intr_dflt: Unassigned channel %d interrupt, disabled

lvluintr: unacknowledged user interrupt (0x%x)

map_wndw: mapping to invalid page %d

map_wndw: mapping to main memory address 0x%x

pq_intr: pq_recv reports no message?

proc_init: invalid compliance level %d (????)

proc_init: invalid config value %d

unmap_wndw: mapping to invalid page %d

user_intr_dflt: Unassigned channel %d interrupt, disabled

userintr: Global error (0x%x)

userintr: unacknowledged user interrupt (0x%x)
```

8.3.36 MIOP EGOS Messages

The following messages originate with the Multibus IOP (MIOP) EGOS. These messages may be caused by hardware failures of the IOP or controller as well as software driver problems or bad data passed to the driver from the CPU side of the driver.

`Address Save Reg 0x%x, Effective address 0x%x`

Description of Error: TBD

Cause and Disposition: TBD

```
Bus error after reset - error logging aborted

Device %d interrupt. Interrupt disabled
```

`IOP 68000 Cache Parity Error (0x%x), PC=0x%6, SR=0x%4`

`IOP 68000 Memory Parity Error (0x%x). PC=0x%x, SR=0x%x`

`IOP 68000 Multibus Parity Error (0x%x), PC=0x%6, SR=0x%4`

`Multibus cable %d error detected`

`PBUS Controller Error (0x%8): %s %s, PC=0x%x, SR=0x%4`

`dev_util: Unknown option %d`

`free_msg: mbs error`

`iop_dev: Unknown request %d from processor %d`

`s%d error log - cable reset`

8.3.37 MIOP EGOS Messages

The following messages originate with the MIOP EGOS.

`lmap_w_mb: mapping to invalid page %d 4p`

Description of Error: TBD

Cause and Disposition: TBD

`lmap_w_mb: mapping to main memory address 0x%x`

`map_w_mb: mapping to invalid page %d`

`map_w_mb: mapping to main memory address 0x%x`

`unmap_w_mb: unmapping invalid window %d`

8.3.38 IOP Internal Errors

The following messages originate with the MIOP and indicate an internal error most likely caused by failed MIOP hardware or corrupted device driver software.

`%s%d error log - cable reset`

Description of Error: TBD

Cause and Disposition: TBD

```
- Slot %d, Src=%s, Flg:0x%b

0x%3", Cache_dbits[flags]

Address Save Reg 0x%x, Effective address 0x%x

Address=0x%5, Access=%s, Source=%s

Buf %d dirty:

Buf %d tag: 0x%b offset=0x%3 (raw_tag=0x%3)

Bus error after reset - error logging aborted

Bus error after reset - error logging aborted

Cache Controller Error (0x%x, 0x%x): PC=0x%x, SR=0x%4

Error(s): 0x%b

PBUS Controller Error (0x%8): %s %s, PC=0x%x, SR=0x%4

VME cable %d log

Window 0x%x, Map register=0x%x (0x%x)

cable_err.c: %s%d error log - cable reset

create_process: Device %d, Illegal call

create_process: Invalid priority %d

create_process: Proc table full!

create_process: p_page allocate failed!

dev_util: Unknown option %d

fork: Illegal call

fork: P_page allocate failed!

fork: Proc table full!

fork: dev=%d, p_estack alloc failed

pdd_init: Idle process p_page alloc failed

pdd_init: struct p_page_def too large

pdd_pstart: Dev %d, number %d, process returned
```

8.3.39 VIOP Internal Errors

The following message originates with the IOP side (CCU side) of an unknown driver.

`Device %d interrupt. Interrupt disabled`

Description of Error: TBD

Cause and Disposition: TBD

`Non-existent VME chassis/cable error`

`VIOP Cache ParErr (0x%x), ADDR=0x%x PC=0x%6, SR=0x%4`

`VIOP Local Mem ParErr (0x%x). ADDR=0x%x PC=0x%x, SR=0x%x`

`VIOP VME ParErr (0x%x), ADDR=0x%x PC=0x%6, SR=0x%4`

`VME cable %d error`

`free_msg: MBS error`

`if (ccu_debuglvl >= lvl)`

`iop_dev: Unknown request %d from processor %d`

`lmap_w_vb: mapping invalid window %d`

`map_w_vb: mapping invalid window %d`

`map_w_vb: mapping to main memory address 0x%x`

`unmap_w_vb: unmapping invalid window %d`

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 9
SPU UNIX Messages

9.1 Overview

This chapter lists SPU UNIX panic messages with a definition, cause, and disposition for each.

A flow chart detailing the process of troubleshooting machine malfunctions based on SPU UNIX panic messages is also included.

9.2 Troubleshooting SPU UNIX Error Messages Flowchart

The following flow chart details the typical sequence of events for isolating failed Field Replaceable Units (FRUs) based on SPU UNIX error messages.

TBD

9.3 SPU UNIX Messages

The following messages are displayed by SPU UNIX when it encounters a problem with which it cannot cope.

`SPU_UNIX(1):pq_init: Processor queue error (head='H', last='L', indx='I')`

Description of Error: During a call to the pq_init routine, an error was detected within the queue structure.
'H' is a variable(hex) that contains the index of the head of the queue.
'L' is a variable(hex) that contains the index of the tail of the queue.
'I' is a variable(hex) that contains the index of the SPU's messages.

Cause and Disposition: This error is caused by a queue that resides in main memory being inconsistent. This could be due to any one of the processors (CPU/CCUs/SPU) overwriting this queue. It also could be due to the queue not being correctly initialized.

`SPU_UNIX(2):pq_recv: Queue head index invalid. (head='H')`

Description of Error: The head index of the current queue is larger than that possible for the system.
'H' is a variable(hex) that represents the invalid head index.

Cause and Disposition: This error is caused by a queue that resides in main memory being inconsistent. This could be due to any one of the processors (CPU/CCUs/SPU) overwriting this queue. It also could be due to the queue not being correctly initialized.

`SPU_UNIX(3):pq_recv: Not in queue or Not owner. (flg='F', own='O')`

Description of Error:
'F' is a variable(hex) that represents the status flag of the message.
'O' is a variable(hex) that represents the owner processor ID for the message.

Cause and Disposition: This error is caused by a queue that resides in main memory being inconsistent. This could be due to any one of the processors (CPU/CCUs/SPU) overwriting this queue. It also could be due to the queue not being correctly initialized.

`SPU_UNIX(4):pq_send: Send to invalid processor id 'I'`

Description of Error: A process on the SPU is trying to send a request to a processor that has a processor number higher than the maximum allowed in this system.
'I' is the processor number of the target processor.

Cause and Disposition: This is probably due to errant user software running on the SPU.

`SPU_UNIX(5):pq_send: Message free or not owner (flg='X' own='O')`

Description of Error: A message is trying to be sent that does not belong to this processor or message is still shown as free.
'F' is a variable(hex) that represents the status flag associated with this message.
'O' is a variable(hex) that represents the message owner processor number.

Cause and Disposition: Unknown

`SPU_UNIX(6):pq_send: Dest proc 'P' last index invalid ('I')`

Description of Error: A message is transmitted that has an incorrect value associated with its last index. The value is greater than the maximum allowed in the system.
'P' is a variable(hex) that represents the destination processor id.
'I' is a variable(hex) that represents the invalid index.

Cause and Disposition: This error is caused by a queue that resides in main memory being inconsistent. This could be due to any one of the processors (CPU/CCUs/SPU) overwriting this queue. It also could be due to the queue not being correctly initialized.

`SPU_UNIX(7):pq_fm_alloc: message 'M' allocated, not free`

Description of Error: A message is allocated from the pool of free messages. When the status of this message is checked, it is not marked free.
'M' is a variable(hex) that represents the message index.

Cause and Disposition: This error is caused by a queue that resides in main memory being inconsistent. This could be due to any one of the processors (CPU/CCUs/SPU) overwriting this queue. It also could be due to the queue not being correctly initialized.

`SPU_UNIX(8):pq_fm_free: freeing free message 'I'`

Description of Error: A message is returned to the free list that is already marked free.
'I' is a variable(hex) that represents the message index.

Cause and Disposition: This is probably due to errant user software.

`SPU_UNIX(9):pq_fm_free: free message list locked!`

Description of Error: A message is returned to the free list and the free list is locked. After waiting for some time, the list remains locked and the attempt is aborted.

Cause and Disposition: This error is caused by a queue that resides in main memory being inconsistent. This could be due to any one of the processors (CPU/CCUs/SPU) overwriting this queue. It also could be the queue not being correctly initialized.

`SPU_UNIX(10):panic:blkdev`

Description of Error: A call to the UNIX routine *getblk* has been made with a major number greater than supposedly possible for SPU UNIX

Cause and Disposition: The cause of this error is unknown. Reboot SPU UNIX, if it occurs repeatedly, replace SPU.

`SPU_UNIX(11):panic:devtab`

Description of Error: When trying to reference the buffer associated with a given block I/O device, a null pointer is referenced.

Cause and Disposition: The cause of this error is unknown. The pointer could have been corrupted by errant system software or a SPU memory problem. Reboot SPU UNIX, if it occurs repeatedly, replace SPU.

`SPU_UNIX(12):panic:IO err in swap`

Description of Error: When trying to swap (in or out) of memory to disk, an I/O error occurred.

Cause and Disposition: This message should be preceded by some type of I/O errors emanating from the disk driver. The I/O error could be helpful in finding this problem. A bad swap partition could be the cause.

`SPU_UNIX(13):physio: address wraparound error`

Description of Error: When performing a transfer to/from an I/O device, the number of bytes requested is so large that the transfer address wraps back around to kernel memory space.

Cause and Disposition: This is probably caused by an errant user program trying to do raw I/O with an extremely large number of bytes requested in the data transfer. If this error occurs frequently, try to determine the data transfer and what processes are running at the time of the error.

`SPU_UNIX(14):physio: transfer address out-of-bounds`

ts='TS' ds='DS' nb='NB' limit='L'

Description of Error: This error is cause by a request to perform an I/O transfer that will access an invalid address.
'Ts' is a variable(hex) that is the end address of the text segment.
'Ds' is a variable(hex) that is the end address of the data segment.
'Nb' is a variable(hex) that is the base address for I/O transfers.
'L' is a variable(hex) that is the remaining number of bytes in the current I/O request.
All addresses are logical addresses.

Cause and Disposition: This is probably caused by an errant user program with either an invalid start transfer address or an invalid number of bytes requested. If this error occurs frequently, try to determine the processes running at this time.

`SPU_UNIX(15):panic:ccucmd: can't receive message`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(16):ct(D): illegal transfer/partition, block='B', cnt='C'`

Description of Error: A request was made to transfer data specifying an invalid partition or for more blocks than is available for a given partition.
'D' is a variable(dec) that is the minor number of the device.
'B' is the requested block number.
'C' is the number of bytes requested.

Cause and Disposition: This error is probably due to a user program error or a bad/corrupt entry in the directory */dev*.

`SPU_UNIX(17):ct: ctlr busy during ctstart (fdc='X'), timeout = 'T'`

Description of Error: During the initiation of a command to the cartridge tape controller, the controller was busy, most likely completing a previous command.
'X' is a variable(hex) that contains the status of the floppy tape control register.
'T' is a variable(dec) that contains the timeout value remaining when the controller de-asserted the busy signal.

Cause and Disposition: This error is not cause for immediate concern. Under most circumstances, recovery will be automatic.

`SPU_UNIX(18):ct: ctlr timed out waiting for idle (fdc='X'), resetting...done`

Description of Error: This error occurs when the controller will not go idle on its own. A 4- to 5-second timeout value is used to wait for the controller to become ready. If it does not become ready during this time, this message will occur and the controller will become reset.
'X' is a variable(hex) that contains the contents of the tape control register when the timeout occurred.

Cause and Disposition: The cause of this error is a cartridge tape controller that does not become ready. This error is probably indicative of a potential SPU hardware problem.

`SPU_UNIX(19):ct: ctlr busy after reset (fdc='X')`

Description of Error: This message will be associated with error number 18. After the timeout occurs (waiting for the controller to go idle), the controller is reset. If the controller is still not ready, this message is displayed.
'X' is a variable(hex) that contains the contents of the tape control register when the timeout occurred.

Cause and Disposition: This error message is most likely caused by a bad SPU.

`SPU_UNIX(20):ct(D): Timeout limit exceeded 'TAPEERROR3'`

Description of Error: A command issued to the cartridge tape controller did not complete within the expected amount of time.
'D' is a variable(dec) that is the minor number of the device.

Cause and Disposition: This error is probably caused by a hardware problem with the SPU or the cartridge tape drive unit.

`SPU_UNIX(21):ct('D'):Bad block table overflow 'TAPEERROR3'`

Description of Error: There are more bad blocks detected on the tape than there is room for in the bad block table.
'D' is a variable(dec) that is the minor number of the device.

Cause and Disposition: This is probably caused by a bad cartridge tape, by dirty heads, or by other problems with the tape drive. Try a new tape first, clean the heads second, replace the drive third.

`SPU_UNIX(22):ct(D):68000 Bus error (X) 'TAPEERROR3'`

Description of Error: This error is caused when a bus error occurs during a cartridge tape I/O operation.
'D' is a variable(dec) that is the minor number of the device.

Cause and Disposition: The cause of this error is unknown.

`SPU_UNIX(23):ctgetaddr: failed on memory copy 'D' 'D'.`

Description of Error: An abnormal condition was detected by either the copyin (D=1) or the copyout (D=2) functions.

Cause and Disposition: The cause of this error is unknown.

`SPU_UNIX(24):ct('D'): 'D'): Seek error detected on read address, trk='T'`

Description of Error: A seek to a specified track was not successful.
'T' is a variable(dec) that is the track where the seek error is detected.
'D' is a variable(dec) that is the minor number of the device.

Cause and Disposition: This could be either a hardware error with the SPU or the cartridge tape driver or could be caused by errant user code trying illegal transfers.

`SPU_UNIX(25):ct('D'): Cartridge tape write protected!`

Description of Error: A write to a write-protected cartridge tape was attempted.
'D' is a variable(dec) that is the minor number of the device.

Cause and Disposition: The tape needs to be removed, the write-protection turned off, then the operation tried again. This could also be caused by a faulty or misadjusted write-protect detection mechanism in the tape drive.

`SPU_UNIX(26):ct('D'): Errors on bad block table 'B0' [,'B1','B2'...]`

Description of Error: Errors were found when reading the specified bad blocks. A list of the blocks are given.
'D' is a variable(dec) that is the minor number of the device.

Cause and Disposition: This is probably caused by a bad cartridge tape (the bad block table has become corrupted). Reformat a new tape and try again. If this occurs often, clean the heads on the drive or replace it.

`SPU_UNIX(27):dk('D'): illegal transfer, block='B', cnt='C'`

Description of Error: An illegal transfer request was made. The error is detected when either an invalid partition is referenced or a block number that is too large to be on the specified device is requested.
'D' is a variable(dec) that is the minor number of the device.
'B' is a variable(dec) that is the block number.
'C' is a variable(dec) that is the number of bytes requested.

Cause and Disposition: This is probably caused by an errant user program. It could also be caused by a corrupt entry in the */dev* directory.

`SPU_UNIX(28):dk('D'): Transfer timeout error, retries = 'R'`

Description of Error: The requested command did not complete within the expected amount of time.
'D' is a variable(dec) that is the minor number of the device.
'R' is a variable(dec) that is the number of times the command was attempted before aborting.

Cause and Disposition: See expanded list and discussion of disk errors below.

`SPU_UNIX(29):dk 68000 bus error ('X'), retries = 'R'`

Description of Error: A bus error occurred during a disk transfer operation.
'D' is a variable(dec) that is the minor number of the device.
'X' is a variable(hex) that is the address where the bus error was detected.
'R' is a variable(dec) that is the number of times the command was attempted before aborting.

Cause and Disposition: The cause of this error is generally unknown. The expanded error report for disk errors (below) could give some answers.

`SPU_UNIX(30):('D'): unexpected sasierror = 'U', retries = 'R'`

Description of Error:
'D' is a variable(dec) that is the minor number of the device.
'U' is a variable(dec) that is the error number returned for this transfer.
'R' is a variable(dec) that is the number of times the command was attempted before aborting.

Cause and Disposition: Unknown

`SPU_UNIX(31):dk('D'): Error status='E' ('S'), retries='R'`

Description of Error: An error was returned from the disk via the SCSI bus during some operation. The cause of the error is described in the message.
'D' is a variable(dec) that is the minor number of the device.
'R' is a variable(dec) that is the number of times the command was attempted before aborting.
'E' is a variable(hex) that is the error status read from the SCSI bus.
'S' is a variable(string) that contains a textual description of the error code.

The possible values of strings and error codes are:

0x00:No Error Status
0x01:No Index/Sector Signal
0x02:No Seek Complete
0x03:Write Fault
0x04:Drive not ready
0x05:Drive not selected
0x06:No Track00
0x10:ID CRC error
0x11:Uncorrectable data error
0x12:ID Address Mark not found
0x13:No address mark in data field
0x14:Record not found
0x15:Seek Error
0x18:Data Check in no retry mode
0x19:ECC error during verify
0x19:Defect list error
0x1a:Interleave error
0x1a:Parameter overrun
0x1c:Unformatted or bad format on drive
0x1c:Primary defect list not found
0x1d:Compare error
0x20:Illegal Command
0x21:Illegal block address
0x22:Illegal function for device type
0x23:Volume overflow
0x24:Bad argument
0x25:Invalid logical unit number
0x26:Invalid field in parameter list
0x27:Write protected
0x28:Media changed
0x29:Device was reset
0x2a:Mode select parameter chaneged
0x2c:Error count overflow
0x31:Medium format corrupted
0x40:Ram failure
0x43:Message reject error
0x44:Internal controller error
0x45:Select/reselect failed
0x47:Scsi interface parity error
0x48:Initiator detected error
0x49:Inappropriate/illegal message

Cause and Disposition: The causes of these errors are either the SPU or the appropriate disk drive or disk drive controller. For expanded information regarding interpretation of these error codes, refer to the manual for the controller/drive that is being used by the SPU.

Disk errors: For errors 28–31 a dump of the command will be sent to the screen. This dump looks like:

'S1' device 'MJ'/'MR' block 'B' XX XX XX XX XX XX

'S1' is a variable(string) that is either "Reading" or "Writing"
'MJ' is a variable(dec) that is the major number of the device.
'MR' is a variable(dec) that is the minor number of the device.
'B' is a variable(dec) that is the block number.
The string of 6 hex digits (XX XX XX XX XX XX) is the actual hex representation of the sasi command that was issued.

`SPU_UNIX(32):qic('D') Byte count not multiple of block size - trailing bytes ignored`

Description of Error: A command (read/write) was issued to the cartridge tape drive with a block size that is not a multiple of the inherent block size (512 bytes) of the drive.
The 'D' variable(dec) is the minor number of the device.

Cause and Disposition: The cause is probably errant user software. The transfer will complete using a truncated block size.

`SPU_UNIX(33):qic('D'): Transfer timeout error`

Description of Error: The requested command did not complete within the expected amount of time.
'D' is a variable(dec) that is the minor number of the device.

Cause and Disposition: Refer to the expanded list and discussion of QIC errors below.

`SPU_UNIX(34):qic('D'): 68000 bus error ('X')`

Description of Error: A bus error occurred during a QIC tape transfer operation.
'D' is a variable(dec) that is the minor number of the device.
'X' is a variable(hex) that is the address where the bus error was detected.

Cause and Disposition: The cause of this error is unknown.

`SPU_UNIX(35):qic('D'): unexpected sasierror = 'U'`

Description of Error: Unknown
'D' is a variable(dec) that is the minor number of the device.
'U' is a variable(dec) that is the error number returned for this transfer.

Cause and Disposition: Unknown

Tape errors: For errors 33–35 a dump of the command will be sent to the screen. This dump appears in the following format:

'S1' device 'MJ'/'MR' block 'B' XX XX XX XX XX XX

'S1' is a variable(string) that is either "Reading" or "Writing."
'MJ' is a variable(dec) that is the major number of the device.
'MR' is a variable(dec) that is the minor number of the device.
'B' is a variable(dec) that is the block number.

The string of 6 hex digits (XX XX XX XX XX XX) is the actual hex representation of the SCSI command that was issued.

`SPU_UNIX(36):qic('D'): Error status='E'`

Description of Error: An error was returned from the QIC tape drive via the SCSI bus during some operation. The cause of the error is described in one of the messages below (37–50).
'D' is a variable(dec) that represents the minor number of the device.
'E' is a variable(hex) that represents the returned SCSI error code.
The error code is the code available to a user program through the *iocntl* call. The possible values of the error codes are:

HARDWARE_ERROR 0x1
MEDIUM_ERROR 0x2
RECOVERED_ERROR 0x4
BLANK_CHECK 0x8
VOLUME_OVERFLOW 0x10

Cause and Disposition: Unknown

`SPU_UNIX(37):qic('D'):recovered error`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is the minor number of the device.

Cause and Disposition: The last command completed successfully following recovery actions by the tape driver. A check condition was not issued.

`SPU_UNIX(38):qic('D'):not ready`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The tape drive cannot be accessed. Operator intervention may be required to correct this condition.

`SPU_UNIX(39):qic('D'):medium error`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The command terminated with a non-recoverable error that was probably caused by a flaw in the medium or an error in the recorded data.

`SPU_UNIX(40):qic('D'):hardware error`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The tape drive detected a non-recoverable hardware failure (parity, etc.) while performing the command.

`SPU_UNIX(41):qic('D'):illegal request - ignored`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The command descriptor block contained an illegal parameter. Probably a SPU UNIX software problem.

`SPU_UNIX(42):qic('D'):unit attention`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The tape drive has been reset or the cartridge has been changed.

`SPU_UNIX(43):qic('D'):data protect`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The cartridge is write-protected (**SAFE**) when a write operation is attempted.

`SPU_UNIX(44):qic('D'):blank check`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: A no-data condition was encountered on the tape.

`SPU_UNIX(45):qic('D'):vendor unique error - ignored`

Description of Error: This message should not be seen since the drive supposedly will not issue it. If it is seen, the tape drive is probably bad.
'D' is a variable(dec) that contains the minor number of the device.

`SPU_UNIX(46):qic('D'):copy aborted error - ignored`

Description of Error: This message should not be seen since the drive supposedly will not issue it. If it is seen, the tape drive is probably bad.
'D' is a variable(dec) that contains the minor number of the device.

`SPU_UNIX(47):qic('D'):aborted command error - ignored`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The tape drive aborted the command. The initiator may be able to recover by trying the command again.

`SPU_UNIX(48):qic('D'):volume overflow`

Description of Error: This error occurs when the QIC drive returns this status code in response to a check status command.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: The tape has reached the physical end-of-medium and data remains in the buffer that has not been written to the tape. Probably user program error.

`SPU_UNIX(49):qic('D'):data miscompare error - ignored`

Description of Error: This message should not be seen since the drive supposedly will not issue it. If it is seen, the tape drive is probably bad.
'D' is a variable(dec) that contains the minor number of the device.

`SPU_UNIX(50):qic('D'):reserved error code 0x0c - ignored`

Description of Error: This message should not be seen since the drive supposedly will not issue it. If it is seen, the tape drive is probably bad.
'D' is a variable(dec) that contains the minor number of the device.

`SPU_UNIX(51):qicioctl: unknown ioctl cmd='C'`

Description of Error: An ioctl system call was issued to the QIC device driver with an unknown command code.
'D' is a variable(dec) that contains the minor number of the device.

Cause and Disposition: This is probably caused by errant user code.

`SPU_UNIX(52):sasibusytest: timed out waiting for idle (cp='X'), resetting...done`

Description of Error: The SCSI bus took longer than allocated to become non-busy (go idle). The SCSI bus is then reset to try to get the bus to become idle.
'X' is a variable(hex) that contains the results of the SPU SCSI status register.

Cause and Disposition: This error is could be caused by either a bad SPU or bad device on the SCSI bus that is holding onto the busy signal.

`SPU_UNIX(53):sasibusytest: SASI busy after reset (cp='S')`
`SPU_UNIX(53):panic:sasibusytest: SASI controller busy after reset`

Description of Error: If after the reset, the bus is still busy, it is assumed that the hardware is in some way broken and a panic occurs.
'S' is a variable that contains the status of the SCSI controller register on the SPU after the timeout.

Cause and Disposition: This is probably caused by some I/O device on the SCSI being broken, a cabling problem or a bad SPU.

`SPU_UNIX(54):sasinitiate: timed out waiting for busy`

Description of Error: The SCSI bus took longer than expected to go busy after the issuance of a command on the bus. The bus is then reset and the command aborted.

Cause and Disposition: The cause of this is probably a device hanging on the SCSI bus or a bad SPU. Software can tolerate this error, so if it only happens occasionally then it is benign.

`SPU_UNIX(55):panic:sasitimeout Sasi_busy`

Description of Error: A condition in which the sasitimeout routine was called and the SCSI bus was not allocated to a device by the SPU.

Cause and Disposition: Unknown

`SPU_UNIX(56):sasiinit(): SASI controller busy! control port = 'S'`

Description of Error: A condition exists in which the SCSI bus is initialized and the bus shows busy during the initialization. This causes any operation pending on the bus to be aborted.
'S' is a variable that contains the status of the SPU's SCSI status register.

Cause and Disposition: Probably a device on the SCSI bus is hanging onto the busy line. This message will often be seen after error #52.

`SPU_UNIX(57):panic:ioccom canq`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(58):panic:ttyrub`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(59):ua overrun`

Description of Error: A received data overrun has been detected on the UART.

Cause and Disposition: Data is coming in too fast for the SPU to handle. This error is often seen when running the system at reduced margins where the data rate relative to the SPU's clock increases.

`SPU_UNIX(60):ua overrun`

Description of Error: A received data overrun has been detected on the UART.

Cause and Disposition: Data is coming in too fast for the SPU to handle. This error is often seen when running the system at reduced margins where the data rate relative to the SPU's clock increases.

`SPU_UNIX(61):DSR FALSE--ignored 'C' from #'D'`

Description of Error: The Data Set Ready (DSR) line has been dropped on one of the UARTs.
'C' is a variable(hex) that is the dropped character.
'D' is the uart number where DSR was de-asserted.

Cause and Disposition:
Check the terminal/modem and cabling connected to this port.

`SPU_UNIX(62):wdioctl: No free alloc table entries`

Description of Error: A window map for the current process could not be allocated. Since there is a one to one correspondence between window maps and available processes, this should not happen.

Cause and Disposition: Unknown. UNIX is corrupted.

`SPU_UNIX(63):wdioctl: PID error (exp='E', act='A')`

Description of Error: During a call to the window *ioctl* handler, a window map was allocated that did not belong to the current process.
'E' is a variable(dec) that represents the expected value of the process id (the current process).
'A' is a variable(dec) that represents the actual value of the process id for this window.

Cause and Disposition: This could be caused if a user program tries to do access through a window that has not been allocated yet.

`SPU_UNIX(64):Bad free count on dev 'MJ' 'MN'`

Description of Error: The free count on the specified disk is greater than the allowed maximum.
'MJ' is a variable(dec) that is the major number of the device.
'MN' is a variable(dec) that is the minor number of the device.

Cause and Disposition: The superblock is probably corrupted on the disk. Run *fsck* and try again.

`SPU_UNIX(65):No space on device 'MJ' 'MN'`

Description of Error: The specified device has no more usable free space.
'MJ' is a variable(dec) that is the major number of the device.
'MN' is a variable(dec) that is the minor number of the device.

Cause and Disposition: The device is probably full; remove some files. The filesystem could also be corrupted, run *fsck*.

`SPU_UNIX(66):Bad block on device 'MJ'/'MN'`

Description of Error: A requested block is outside of the range from the end of the ilist to the size of the device.
'MJ' is a variable(dec) that is the major number of the device.
'MN' is a variable(dec) that is the minor number of the device.

Cause and Disposition: A bad file system could have been mounted. Or the file system could be corrupted.

`SPU_UNIX(67):Out of inodes on device 'MJ'/'MN'`

Description of Error: There are no more inodes available on the specified device.
'MJ' is a variable(dec) that is the major number of the device.
'MN' is a variable(dec) that is the minor number of the device.

Cause and Disposition: Could be too many files in existence; delete some files. Filesystem could be corrupted; run *fsck*.

`SPU_UNIX(68):Bad count on device 'MJ'/'MN'`

Description of Error: Either the number of inodes available or the number of free blocks available is greater than the allowable maximum. Both of these values will then get set to zero, which will cause the no space message to be issued during the next transfer.
'MJ' is a variable(dec) that is the major number of the device.
'MN' is a variable(dec) that is the minor number of the device.

Cause and Disposition: Probably a corrupted filesystem. Run *fsck*.

`SPU_UNIX(69):panic:No fs`

Description of Error: The device for which a request is being made is not in the mount table.

Cause and Disposition: UNIX is corrupted. Reboot.

`SPU_UNIX(70):panic:Timeout table overflow`

Description of Error: During a call to the timeout function, there is no space in the callout structure to place the current request. There is nothing intelligent to be done if an entry will not fit, thus the panic.

Cause and Disposition: Probably a corrupted UNIX. Reboot and try again.

`SPU_UNIX(71):Detected illegal line clock of 'H' Hz, using 60 Hz`

Description of Error: SPU UNIX tries to determine the line frequency to know if it is running on a 60 Hz or 50 Hz system. If the measured value is not close to either 50 Hz or 60 Hz, this message will appear.
'H' is a variable that contains the measured clock frequency

Cause and Disposition: The cause of this could be a bad SPU timer chip which is used to calculate the line frequency. Also a bad SCM could be causing a low quality clock signal to reach the SPU. The electrical power utility serving the site could be having trouble with their generators, although this is unlikely.

`SPU_UNIX(72):No detectable line clock -- using 60 Hz`

Description of Error: During the time that the SPU is trying to measure the line clock frequency, no clock is detected.

Cause and Disposition: This could be caused by a bad timer chip on the SPU or a poor quality signal being generated by the System Control Module (SCM). If there truly is no line clock, SPU UNIX will hang shortly after this message appears.

`SPU_UNIX(73):Falloc:No file`

Description of Error: During a call to the *falloc* routine to allocate a file descriptor, no file desciptors/structures are available.

Cause and Disposition: This message is usually no cause for alarm. Generally, this message appears when too many files have been opened by one of the processes currently executing on the SPU. Try to determine which process has opened the excessive number of files.

`SPU_UNIX(74):panic:Fs not in mount table`

Description of Error: An inode is mounted, but the mounted file system is not found in the mount table.

Cause and Disposition: UNIX is corrupted. Reboot and try again.

`SPU_UNIX(75):Inode table overflow`

Description of Error: Too many inodes are currently in use (in core).

Cause and Disposition: Too many processes are running with too many inodes in use. The problem should be self-correcting. Check which processes are running.

`SPU_UNIX(76):Iaddress > 2^24`

Description of Error: A block address referenced inside an inode structure is $> 2^{24}$. This message is for information purposes only, since it appears that no other action is taken when this message occurs.

Cause and Disposition: The in-core copy of the inode is probably corrupt in some manner. Reboot.

`SPU_UNIX(77):Link error. &ftrap = 'X'fR`

OR

`SPU_UNIX(77):Link error. &fault = 'X'`

Description of Error: These errors will occur during booting the system if there is an address alignment error with either the fault or ftrap structures.
'X' is a variable(hex) that represents the address of the structure.

Cause and Disposition: This should only happen if the disk copy of UNIX is corrupted. Try a root restore.

`SPU_UNIX(78): panic:No room for scn_structs`

Description of Error: This is a SP2 message that only occurs during the boot process. There is not enough memory available to hold the structures for the system scan ring descriptions.

Cause and Disposition: Something is probably wrong with the version of SPU UNIX, or the SPU memory system could be broken in such a way that the memory sizing routine determines that there is less available memory than is actually installed.

`SPU_UNIX(79): panic:No room for ring structs`

Description of Error: This is a SP2 message that only occurs during the boot process. There is not enough memory available to hold the structures for the system scan ring descriptions.

Cause and Disposition: Something is probably wrong with the version of SPU UNIX, or the SPU memory system could be broken in such a way that the memory sizing routine determines that there is less available memory than is actually installed.

`SPU_UNIX(80):panic:Iinit can not read superblock`

Description of Error: During the initialization process, an I/O error occurs during the attempted read of the superblock.

Cause and Disposition: An I/O error will probably proceed this message. Take appropriate action based on the results of the I/O error.

`SPU_UNIX(81):Msg_exit: lost message chain from index %d`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(82):Msg_intr: Bus error in step 'S' ADDR='X'`

Description of Error: During the handling of a Message Based System (MBS) related system interrupt (system interrupt 11), a bus error occurred.
'S' is a variable(dec) that represents which step in the code was being executed when the bus error occurred.
'X' is a variable(hex) that represents the address where the bus error was detected.

Cause and Disposition: The cause of this error is generally not known. It is suspected that it could be a bad SPU. However a bad memory system could also cause this error since many of the mbs related functions are accessing main memory.

`SPU_UNIX(83):Msg_intr: Queue locked for 'P' passes. DISABLED`

Description of Error: After the receipt of an interrupt an attempt to de-queue a message fails because the queue remains locked.

Cause and Disposition: Could be an inadvertent interrupt or someone may have destroyed the queue structure in main memory.

`SPU_UNIX(85):Ptrace:Shared text`

Description of Error: A system *ptrace* system call was executed that tried to write into the text segment of a process that was using a shared text segment.

Cause and Disposition: Check for multiple invocations of a given process running. The adb utility is one of the few that use the *ptrace* system call.

`SPU_UNIX(86):Ptrace:usraccess 'X'`

Description of Error: A *ptrace* system call was made that tried to access kernel memory from user space.
'X' is a variable(hex) that contains the address of the attempted access.

Cause and Disposition: Probably errant user software incorrectly using the *ptrace* system call.

`SPU_UNIX(87):Ptrace:suiword failed`

Description of Error: During the execution of a *ptrace* system call, a copy of a word to user space failed.

Cause and Disposition: This is probably caused by an improper address passed to the *ptrace* call or to malfunctioning hardware on the SPU.

`SPU_UNIX(88):Proc on q`

Description of Error: A process is being added to the run queue that already exists on the run queue.

Cause and Disposition: Probably a corrupted UNIX. The process will not be added to the run queue so it may be possible to recover from this error.

`SPU_UNIX(89):panic:Running a dead proc`

Description of Error: A process is set to run that is in a zombie state (exited and waiting for parent to accept status with wait call).

Cause and Disposition: UNIX is corrupted. Reboot

`SPU_UNIX(90):panic:No procs`

Description of Error: During the execution of a *fork* system call a routine detects that there are no slots in the process table available. This is a panic at this point because prior to the point of the panic, the table was checked and there were slots in the process table.

Cause and Disposition: This is a transient UNIX-is-damaged type of error and will probably never appear. But if it does, reboot.

`SPU_UNIX(91):panic:Out of swap`

Description of Error: During the execution of an *exec* system call the system runs out of swap space.

Cause and Disposition: This error should not happen if the system is configured correctly. Could be that the entries in *dkparmtab* for */dev/swap* are corrupted. Try to reboot. If it appears again, do a root restore.

`SPU_UNIX(92):User access to kernel space 'X'`

Description of Error: During the execution of a read or write system call, a pointer is passed to the kernel that is outside of user space. The system call will then be aborted.
'X' is a variable(hex) that represents the address that was being referenced.

Cause and Disposition: This is probably caused by errant user code and can be ignored.

`SPU_UNIX(93):panic:Out of swap space`

Description of Error: During the swapping out of a process, the system runs out of swap space.

Cause and Disposition: This error should not happen if the system is configured correctly. Could be that the entries in *dkparmtab* for */dev/swap* are corrupted. Try to reboot. If it appears again, do a root restore.

`SPU_UNIX(94):Out of text`

Description of Error: During the establishment of the text region for a process, the text descriptor structure overflows.

Cause and Disposition: Probably too many processes are running. Or UNIX's internal structures are corrupted.

`SPU_UNIX(95):panic:Out of swap space`

Description of Error: During the creation of a text segment, the system runs out of swap space.

Cause and Disposition: This error should not happen if the system is configured correctly. Could be that the entries in *dkparmtabr* for */dev/swap* are corrupted. Try to reboot. If it appears again, do a root restore.

`SPU_UNIX(96):Trap: 'TRAPMESSAGE', 'MODE' mode`

Description of Error: These are system execptions that occur during the execution of SPU UNIX. The user mode errors can be safely ignored. The system mode errors, however, will cause a panic.

TRAPMESSAGE is a variable(string) that describes the trap. The values this variable can take on are:

reset initial ssp?	level 1 autovector
reset initial pc?	level 2 autovector
bus error	level 3 autovector
address error	level 4 autovector
illegal instruction	level 5 autovector
zero divide	level 6 autovector
chk instruction	level 7 autovector
trapv instruction	trap #0
privilege violation	trap #1
trace	trap #2
line 1010 emulator	trap #3
line 1111 emulator	trap #4
reserved #1	trap #5
reserved #2	trap #6
reserved #3	trap #7
reserved #4	trap #8
reserved #5	trap #9
reserved #6	trap #A
reserved #7	trap #B
reserved #8	trap #C
reserved #9 trap #D	
reserved #10 trap #E	
reserved #11 trap #F	
reserved #12	
spurious interrupt	

These trap messages have a one to one correspondence with 68000 trap messages. For more information on these trap messages, refer to the *Motorola MC68000 User's Manual.*

'MODE' is a variable (string) that can take on the values "user" "kernel." This variable tells where the system was executing when the trap occurred. A trap will follow (message 98).

Cause and Disposition: The cause should be determined from the message. Kernel messages should not normally occur, could be a sign of bad SPU hardware or a bad copy of the kernel. Reboot. If there are still problems, replace the SPU.

`SPU_UNIX(97):Trap: interrupt vector 'X', 'MODE'`

Description of Error: A trap was taken for either an unassigned reserved vector or a user interrupt vector that was not expected. A trap will follow (message 98).

Cause and Disposition: Probably broken hardware on the SPU.

`SPU_UNIX(98):panic:trap`

Description of Error: This message is issued after either message number 96 or 97.

Cause and Disposition: Refer to message number 96 or 97

`SPU_UNIX(99):Trap: SCM Power Fail Interrupt`

Description of Error: The System Control Module (SCM) detected a power fail condition and sent an interrupt.

Cause and Disposition: If there is actually a power fail condition, the SPU will die immediately after issuing this message. Fix the power problem. If the SPU stays alive then the SCM or the SPU (or the interface) could be broken in such a way that the SPU is detecting these interrupts.

`SPU_UNIX(100):Trap: SPU memory parity error, 'MODE' mode`

Description of Error: A parity error has been detected in the SPU RAM. 'MODE' is a variable (string) that is either "user" or "kernel."

Cause and Disposition: If the parity error occurs in a user process, that process is sent a segmentation violation signal. If the parity error occurs in the kernel then a panic occurs (refer to message 101). If the parity errors are frequent, replace the SPU.

`SPU_UNIX(101):panic:SPU memory parity error in kernel`

Description of Error: A parity error occurred while executing in kernel space.

Cause and Disposition: Refer to message message 100.

`SPU_UNIX(102):Trap: kernel mode bus error`

OR

`SPU_UNIX(102):Trap: kernel mode address error`

Description of Error: A bus or an address error occurred while executing in kernel space. A panic will immediately follow.

Cause and Disposition: This could be caused by faulty hardware or corrupted software. Reboot. If the problem persists, replace the SPU.

`SPU_UNIX(103):panic:Kernel bus/address error`

Description of Error: This panic is caused by error 102.

Cause and Disposition: Refer to message 102.

`SPU_UNIX(104):Trap: spurious interrupt ignored`

Description of Error: The 68000 generated a spurious interrupt trap.

Cause and Disposition: This is caused when a bus error occurs during a 68000 interrupt acknowledge cycle. This is probably harmless unless it occurs often, in which case, replace the SPU.

`SPU_UNIX(105):Catchint:pid='P' signo='S',ix='I'`

Description of Error: A call to *catchint* has been made with a vector greater than 256, which is greater than possible on the SPU.
'P' is a variable(dec) that is the process number of the process making the call.
'S' is a variable(hex) that is the signal number trying to be assigned.
'I' is a variable(dec) that represents the index into an internal array of interrupts that can be caught.

Cause and Disposition: This is caused by errant user code.

`SPU_UNIX(106):Pitchint - invalid vector 'X'`

Description of Error: An interrupt occurred for a potentially catchable interrupt. But when the index of the interrupt is referenced it is not found.
'X' is a variable(hex) that is the 68000 vector.

Cause and Disposition: This is a corrupt UNIX error. Reboot and try again.

Discussion of Trap Messages Resulting from Hardware Interrupts

The following messages (error codes 107–126) are caused by the SPU receiving an interrupt from some hardware source that is normally under the control of a user program. SPU UNIX has a provision for mapping these hardware interrupts into signals through the two system calls *catchint* and *signal*.

Used in conjunction, these system calls allow for a user program to catch the occurrence of some hardware interrupt. In order for one of these interrupts to be accepted by the SPU, the bit in the SPU's interrupt enable register must be set to enable the particular interrupt. If the SPU detects one of these interrupts and no user program has previously tried to "catch" the interrupt via the *catchint* system call, one of these trap messages will be displayed.

Thus, for one of these messages to occur, the hardware must generate the source of the interrupt, the bit in the Interupt Enable Register (IER) corresponding to the interrupt must be set, AND there must be no user program with an outstanding catch on that interrupt.

Getting one of these messages could be caused by faulty user code setting a bit in the IER incorrectly or faulty hardware generating or receiving unexpected interrupts.

`SPU_UNIX(107):Trap: SPU environment error`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(108):Trap: Hard error`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(109):Trap: Memory soft error`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(110):Trap: System interrupt acknowledge`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(111):Trap: System interrupt 15`

Description of Error: System interrupt 15 was received by the SPU when no user program was expecting it.

Cause and Disposition: There are several possibilities for this error. Any processor in the system can interrupt on this channel. An errant program in any of these processors (including the SPU) could send an interrupt to the SPU on this channel. The SPUs interrupt detection logic could be broken, as could any other processor's interrupt generation logic. Most of these messages have been traced to software problems on one of the processors, especially the SPU.

`SPU_UNIX(112):Trap: System interrupt 14`

Description of Error: System interrupt 14 was received by the SPU when no user program was expecting it.

Cause and Disposition: Refer to description for message 111.

`SPU_UNIX(113):Trap: System interrupt 13`

Description of Error: System interrupt 13 was received by the SPU when no user program was expecting it.

Cause and Disposition: Refer to description for message 111.

`SPU_UNIX(114):Trap: System interrupt 12`

Description of Error: System interrupt 12 was received by the SPU when no user program was expecting it.

Cause and Disposition: Refer to description for message 111.

`SPU_UNIX(115):Trap: System interrupt 11`

Description of Error: System interrupt 11 was received by the SPU when no user program was expecting it.

Cause and Disposition: Refer to description for message 111.

`SPU_UNIX(116):Trap: System interrupt 10`

Description of Error: System interrupt 10 was received by the SPU when no user program was expecting it.

Cause and Disposition: Refer to description for message 111.

`SPU_UNIX(117):Trap: System interrupt 09`

Description of Error: System interrupt 9 was received by the SPU when no user program was expecting it.

Cause and Disposition: Refer to description for message 111.

`SPU_UNIX(118):Trap: System interrupt 08`

Description of Error: System interrupt 8 was received by the SPU when no user program was expecting it.

Cause and Disposition: Refer to description for message 111.

`SPU_UNIX(119):Trap: DMA channel 3 normal interrupt`

Description of Error: An interrupt was generated by the DMA channel 3 interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(120):Trap: DMA channel 3 error interrupt`

Description of Error: An interrupt was generated by the DMA channel 3 error interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(121):Trap: DMA channel 2 normal interrupt`

Description of Error: An interrupt was generated by the DMA channel 2 interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(122):Trap: DMA channel 2 error interrupt`

Description of Error: An interrupt was generated by the DMA channel 2 error interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(123):Trap: DMA channel 1 normal interrupt`

Description of Error: An interrupt was generated by the DMA channel 1 interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(124):Trap: DMA channel 1 error interrupt`

Description of Error: An interrupt was generated by the DMA channel 1 error interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(125):Trap: DMA channel 0 normal interrupt`

Description of Error: An interrupt was generated by the DMA channel 0 interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(126):Trap: DMA channel 0 error interrupt`

Description of Error: An interrupt was generated by the DMA channel 0 error interrupt circuitry that was not expected by any user program.

Cause and Disposition: This could be caused by an errant user program or by faulty DMA hardware.

`SPU_UNIX(127):ct('D'):Error:Pos 'TAPEERROR1' 'TAPEERROR3'`

Description of Error: Unknown

Cause and Disposition: Unknown

`SPU_UNIX(128):ct('D'):Error:Data 'TAPEERROR2' 'TAPEERROR3'`

Description of Error: Unknown

Cause and Disposition: (Expanded display for error messages 20, 21, 127, 128, and 22)

In the errors above, there are three variable fields not explained: 'TAPEERROR1', 'TAPEERROR2', and 'TAPEERROR3'. These three fields are described here.

The **'TAPEERROR1'** message consists of one or more messages. Each of these messages corresponds to a bit in the status register of the device. The possible messages and their meaning are:
FifoNtEmpty:
NotRdy:
WrtPrt:
B5:
SeekErr:
CrcErr:
B2:

The **'TAPEERROR2'** message consists of one or more messages. Each of these messages corresponds to a bit in the status register of the device. The possible messages and their meaning are:
FifoNtEmpty:
NotRdy:
WrtPrt:
DelData:
RcdNtFnd:
CrcErr:
LostData:
These strings are all printed separated by blanks on a single line.

The **'TAPEERROR3'** message consists of the following:
cterr='C', retries='R'
ct('D'): stream='S' segment='SG' 'V'='SV'
The 'C' variable(hex) is the error number returned by the low-level cartridge tape routines the possible values and meanings are:
The 'R' variable(dec) is the number of times that the command is tried before the error is reported
The 'D' variable(dec) is the minor number of the device
The 'S' variable(dec) is the stream number where the error occurred
The 'SG' variable(dec) is the segment number where the error was detected
The variable 'V' (string) is either "current segment" or "sector"
The variable 'SV' (dec) is either the sector number or the segment number depending on the value of 'V'

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 10
CONVEX UNIX Messages

10.1 Overview

This chapter lists CONVEX UNIX panic messages with a definition, cause, and disposition for each.

A flow chart detailing the process of troubleshooting machine malfunctions based on CONVEX UNIX panic messages is also included.

CONVEX UNIX panic messages are divided into four categories:

- CONVEX UNIX panic messages
- CONVEX UNIX panic messages from the High-Speed Parallel device (HSP)
- CONVEX UNIX panic messages from the Input/Output Processor (IOP)
- CONVEX UNIX panic messages from the VMEbus Input/Output Processor (VIOP)

Each of these categories is covered in a separate section.

10.2 Troubleshooting CONVEX UNIX Error Messages Flowchart

The following flow chart details the typical sequence of events for isolating failed boards based on CONVEX UNIX error messages.

TBD

10.3 CONVEX UNIX Panic Messages

The following messages are displayed by CONVEX UNIX when it encounters a problem with which it cannot cope.

B_PHYS

Description of Error:

Cause and Disposition:

`Bad mask`

Description of Error:

Cause and Disposition:

`CKU_BUSY can't happen`

Description of Error:

Cause and Disposition:

`CKU_WANTED can't happen`

Description of Error:

Cause and Disposition:

`Invalid PTE fault in Kernel`

Description of Error:

Cause and Disposition:

`Invalid cputype`

Description of Error:

Cause and Disposition:

`Machine check`

Description of Error:

Cause and Disposition:

`No nspcb`

Description of Error:

Cause and Disposition:

`No nspcb in spp_input`

Description of Error:

Cause and Disposition:

`Not enough memory for file system buffers`

Description of Error:

Cause and Disposition:

`Processor queue set up failure`

Description of Error:

Cause and Disposition:

`Vector Valid Fault in Kernel`

Description of Error:

Cause and Disposition:

`bad device name`

Description of Error:

Cause and Disposition:

`bq_free: freeing invalid buffer`

Description of Error:

Cause and Disposition:

`caattach: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`caattach: pq_fm_free error`

Description of Error:

Cause and Disposition:

`cactl: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`cactl: pq_fm_free error`

Description of Error:

Cause and Disposition:

`cactl: tm_send error`

Description of Error:

Cause and Disposition:

`caprobe: impossible error from IOP`

Description of Error:

Cause and Disposition:

`caprobe: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`caprobe: pq_fm_free error`

Description of Error:

Cause and Disposition:

`caprobe: probe msg error "can't happen"`

Description of Error:

Cause and Disposition:

`caprobe: tas byte msg error "can't happen"`

Description of Error:

Cause and Disposition:

`caprobe: tas lock page full`

Description of Error:

Cause and Disposition:

`caprobe: tty flag structure full`

Description of Error:

Cause and Disposition:

`ccucmd: tm_send`

Description of Error:

Cause and Disposition:

`clget: no available clients`

Description of Error:

Cause and Disposition:

`clget: null client`

Description of Error:

Cause and Disposition:

`cnctl free`

Description of Error:

Cause and Disposition:

`cnctl send`

Description of Error:

Cause and Disposition:

`cnopen`

Description of Error:

Cause and Disposition:

`cnopen: premature console open`

Description of Error:

Cause and Disposition:

`conf_via_spu: conf msg invalid`

Description of Error:

Cause and Disposition:

`conf_via_spu: message return error`

Description of Error:

Cause and Disposition:

`console_init: msg alloc error`

Description of Error:

Cause and Disposition:

`console_init: pq_fm_free`

Description of Error:

Cause and Disposition:

`console_init: tas lock page full`

Description of Error:

Cause and Disposition:

`console_init: tm_send`

Description of Error:

Cause and Disposition:

`console_init: tty flag structure full`

Description of Error:

Cause and Disposition:

`copysegn`

Description of Error:

Cause and Disposition:

`create_uzero: arith code`

Description of Error:

Cause and Disposition:

`create_uzero: bkpt_code`

Description of Error:

Cause and Disposition:

`create_uzero: default trap handler size too large`

Description of Error:

Cause and Disposition:

`create_uzero: sig code`

Description of Error:

Cause and Disposition:

`create_uzero: trace code`

Description of Error:

Cause and Disposition:

`daattach: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`daattach: pq_fm_free error`

Description of Error:

Cause and Disposition:

`daclose: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`daclose: pq_fm_free error`

Description of Error:

Cause and Disposition:

`daclose: tm_send error`

Description of Error:

Cause and Disposition:

`dadump: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`dadump: pq_fm_free error`

Description of Error:

Cause and Disposition:

`dadump: tm_send error`

Description of Error:

Cause and Disposition:

`daintr: pq_fm_free error`

Description of Error:

Cause and Disposition:

`daintr: sent count < 0`

Description of Error:

Cause and Disposition:

`daintr: tm_send error`

Description of Error:

Cause and Disposition:

`daioctl: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`daioctl: pq_fm_free error`

Description of Error:

Cause and Disposition:

`daopen: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`daopen: pq_fm_free error`

Description of Error:

Cause and Disposition:

`daopen: tm_send error`

Description of Error:

Cause and Disposition:

`daprobe: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`daprobe: pq_fm_free error`

Description of Error:

Cause and Disposition:

`dastrategy: NULL bp`

Description of Error:

Cause and Disposition:

`dastrategy: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`dastrategy: struct buf done`

Description of Error:

Cause and Disposition:

`ddintr: tm_send error`

Description of Error:

Cause and Disposition:

`ddioctl: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`ddioctl: pq_fm_free error`

Description of Error:

Cause and Disposition:

`ddioctl: tm_send error`

Description of Error:

Cause and Disposition:

`ddstrategy: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`ddstrategy: struct buf done`

Description of Error:

Cause and Disposition:

`ddstrategy: tm_send error`

Description of Error:

Cause and Disposition:

`do_bio: write count < 0`

Description of Error:

Cause and Disposition:

`ex_postrecvbufs: not enough mbufs for initialization`

Description of Error:

Cause and Disposition:

`ex_receive_msg contains NULL mbuf chain pointer`

Description of Error:

Cause and Disposition:

`exinit: no bq message receive buffers`

Description of Error:

Cause and Disposition:

`fhandle and lockhandle-id are not the same size!`

Description of Error:

Cause and Disposition:

`heattach: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`heattach: pq_fm_free error`

Description of Error:

Cause and Disposition:

`heio: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`heio: pq_fm_free error`

Description of Error:

Cause and Disposition:

`heio: tm_send error`

Description of Error:

Cause and Disposition:

`heioctl: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`heioctl: pq_fm_free error`

Description of Error:

Cause and Disposition:

`heioctl: tm_send error`

Description of Error:

Cause and Disposition:

`hyprobe: tas page full`

Description of Error:

Cause and Disposition:

`icmp len`

Description of Error:

Cause and Disposition:

`icmp_error`

Description of Error:

Cause and Disposition:

`idp_usrreq`

Description of Error:

Cause and Disposition:

`if_ikprobe: tas page full`

Description of Error:

Cause and Disposition:

`ikprobe: tas page full`

Description of Error:

Cause and Disposition:

`imemall: out of memory`

Description of Error:

Cause and Disposition:

`in_control`

Description of Error:

Cause and Disposition:

`increase sysgen parameter SYSPTSIZE`

Description of Error:

Cause and Disposition:

`inpcb has no tcpcb`

Description of Error:

Cause and Disposition:

`inpcb has no tcpcb again`

Description of Error:

Cause and Disposition:

`invalid interrupt`

Description of Error:

Cause and Disposition:

`invalid mbuf in recv buffer`

Description of Error:

Cause and Disposition:

`invalid non-resident page fault`

Description of Error:

Cause and Disposition:

`invalid trap code`

Description of Error:

Cause and Disposition:

`ip_init`

Description of Error:

Cause and Disposition:

`ip_stripoptions: corrupted mbuf`

Description of Error:

Cause and Disposition:

`jp_dev: conf end message return`

Description of Error:

Cause and Disposition:

`jp_dev: conf msg overflow`

Description of Error:

Cause and Disposition:

`jp_dev: pq_fm_free fail`

Description of Error:

Cause and Disposition:

`jp_dev: unknown pq_recv return code`

Description of Error:

Cause and Disposition:

`kudp_fastend: mget1`

Description of Error:

Cause and Disposition:

`kudp_fastsend: mget2`

Description of Error:

Cause and Disposition:

`kudp_fastsend: mget3`

Description of Error:

Cause and Disposition:

`lock out of range`

Description of Error:

Cause and Disposition:

`log2`

Description of Error:

Cause and Disposition:

`mapioctl`

Description of Error:

Cause and Disposition:

`msg_intr: error freeing discarded message`

Description of Error:

Cause and Disposition:

`msg_intr: pq_recv returned ERROR`

Description of Error:

Cause and Disposition:

`net_intr: unknown networking interrupt`

Description of Error:

Cause and Disposition:

`nfs_badop`

Description of Error:

Cause and Disposition:

`nfs_strategy: invalid buffer`

Description of Error:

Cause and Disposition:

`nfs_strategy: swapping to nfs`

Description of Error:

Cause and Disposition:

`no bq message receive buffers`

Description of Error:

Cause and Disposition:

`no bq recv bufs available`

Description of Error:

Cause and Disposition:

`no bq recv bufs available???`

Description of Error:

Cause and Disposition:

`no mbufs on slclose`

Description of Error:

Cause and Disposition:

`no memory: Increase SYSPTSIZE`

Description of Error:

Cause and Disposition:

`not enough mbufs for initialization`

Description of Error:

Cause and Disposition:

`ns_err_error`

Description of Error:

Cause and Disposition:

`paattach: kmem_alloc failed`

Description of Error:

Cause and Disposition:

`pagemove`

Description of Error:

Cause and Disposition:

`panic (trap_code[code / 4]);`

Description of Error:

Cause and Disposition:

`panic paprobe: tas page full`

Description of Error:

Cause and Disposition:

`panic(trap_code[ code/4 ]);`

Description of Error:

Cause and Disposition:

`panics or returns EX_SUCCESS`

Description of Error:

Cause and Disposition:

`paprobe: tas page full`

Description of Error:

Cause and Disposition:

`pb: tm_send error`

Description of Error:

Cause and Disposition:

`pb_alloc_msg: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`pbattach: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`pbattach: pq_fm_free error`

Description of Error:

Cause and Disposition:

`pbintr: pq_fm_free error`

Description of Error:

Cause and Disposition:

`pbioctl: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`pbioctl: tm_send error`

Description of Error:

Cause and Disposition:

`pbprobe: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`pbprobe: pq_fm_free error`

Description of Error:

Cause and Disposition:

`pbwrite: b_addr not aligned`

Description of Error:

Cause and Disposition:

`pbwrite: pq_fm_free error`

Description of Error:

Cause and Disposition:

`physreentered`

Description of Error:

Cause and Disposition:

`ptable fault`

Description of Error:

Cause and Disposition:

`pulling B_READ`

Description of Error:

Cause and Disposition:

`raw_attach: socketpeer acquire failed`

Description of Error:

Cause and Disposition:

`raw_usrreq`

Description of Error:

Cause and Disposition:

`revarp: no mbufs`

Description of Error:

Cause and Disposition:

`rtfree`

Description of Error:

Cause and Disposition:

`rwvp: zero size`

Description of Error:

Cause and Disposition:

`set_toc: Wrong processor type`

Description of Error:

Cause and Disposition:

`sp_usrreq`

Description of Error:

Cause and Disposition:

`spp_output REXMT`

Description of Error:

Cause and Disposition:

`spu_close: message alloc error`

Description of Error:

Cause and Disposition:

`spu_close: message free error`

Description of Error:

Cause and Disposition:

`spu_ioctl: message alloc error`

Description of Error:

Cause and Disposition:

`spu_ioctl: message free error`

Description of Error:

Cause and Disposition:

`spu_rw: message alloc error`

Description of Error:

Cause and Disposition:

`spu_rw: message free error`

Description of Error:

Cause and Disposition:

`spuopen: can't happen`

Description of Error:

Cause and Disposition:

`spuopen: message alloc error`

Description of Error:

Cause and Disposition:

`spuopen: message free error`

Description of Error:

Cause and Disposition:

`startrtclock: Unknown cpu type`

Description of Error:

Cause and Disposition:

`startup: bad cmap size`

Description of Error:

Cause and Disposition:

`stbreakup: nbytes non zero on exit`

Description of Error:

Cause and Disposition:

`stbreakup: request extends beyond end of device`

Description of Error:

Cause and Disposition:

`stdone: b_back linked by disk driver`

Description of Error:

Cause and Disposition:

`stdone: bp already done`

Description of Error:

Cause and Disposition:

`stqueue b_resid`

Description of Error:

Cause and Disposition:

`stqueue: empty queue`

Description of Error:

Cause and Disposition:

`stread: bad minor device number`

Description of Error:

Cause and Disposition:

`ststrategy: already done`

Description of Error:

Cause and Disposition:

`ststrategy: bad minor device number`

Description of Error:

Cause and Disposition:

`ststrategy: empty queue`

Description of Error:

Cause and Disposition:

`sys pt too small`

Description of Error:

Cause and Disposition:

`syscall`

Description of Error:

Cause and Disposition:

`syscall: A process running and enqueued`

Description of Error:

Cause and Disposition:

`syscall: B process running and enqueued`

Description of Error:

Cause and Disposition:

`sysgen parameter SYSPTSIZE too small for configuration`

Description of Error:

Cause and Disposition:

`taprobe: tas page full`

Description of Error:

Cause and Disposition:

`tcp_output`

Description of Error:

Cause and Disposition:

`tcp_output REXMT`

Description of Error:

Cause and Disposition:

`tcp_pulloutofband`

Description of Error:

Cause and Disposition:

`tcp_stripoptions: corrupted mbuf`

Description of Error:

Cause and Disposition:

`tcp_usrreq`

Description of Error:

Cause and Disposition:

`tm_intr: invalid action defined`

Description of Error:

Cause and Disposition:

`tm_intr: message mismatch`

Description of Error:

Cause and Disposition:

`tm_send error`

Description of Error:

Cause and Disposition:

`trap: A process running and enqueued`

Description of Error:

Cause and Disposition:

`trap: B process running and enqueued`

Description of Error:

Cause and Disposition:

`udcmd: tm_send`

Description of Error:

Cause and Disposition:

`udiprobe: tm_send to IOP processor device failed`

Description of Error:

Cause and Disposition:

`udireprobe: tm_send to IOP processor device failed`

Description of Error:

Cause and Disposition:

`udp_usrreq`

Description of Error:

Cause and Disposition:

`udprobe: tm_send to IOP processor device failed`

Description of Error:

Cause and Disposition:

`udreprobe: tm_send to IOP processor device failed`

Description of Error:

Cause and Disposition:

`udvprobe: tm_send to VIOP processor device failed`

Description of Error:

Cause and Disposition:

`udvreprobe: tm_send to VIOP processor device failed`

Description of Error:

Cause and Disposition:

`unrecognized interrupt`

Description of Error:

Cause and Disposition:

`update_time: Wrong processor type`

Description of Error:

Cause and Disposition:

`vector fault owner`

Description of Error:

Cause and Disposition:

`vregall: invalid cpu affinity`

Description of Error:

Cause and Disposition:

`vregfree: Invalid cpu affinity`

Description of Error:

Cause and Disposition:

10.4 CONVEX UNIX/HSP Panic Messages

The following messages are displayed by CONVEX UNIX when it encounters a problem in the HSP with which it cannot cope.

`ccuopen: can't get windows`

Description of Error:

Cause and Disposition:

`ccurw bcount`

Description of Error:

Cause and Disposition:

`ccurw no ccu_win`

Description of Error:

Cause and Disposition:

10.5 CONVEX UNIX/IOP Panic Messages

The following messages are displayed by CONVEX UNIX when it encounters a problem in the IOP with which it cannot cope.

`Bad mbuf chain in netmsg`

Description of Error:

Cause and Disposition:

`ccuopen: can't get windows`

Description of Error:

Cause and Disposition:

`ccurw bcount`

Description of Error:

Cause and Disposition:

`ccurw no ccu_win`

Description of Error:

Cause and Disposition:

`dastart: cl == 0`

Description of Error:

Cause and Disposition:

`empty receive buf`

Description of Error:

Cause and Disposition:

`invalid mbuf in recv buffer`

Description of Error:

Cause and Disposition:

`lock out of range`

Description of Error:

Cause and Disposition:

`msg alloc failure in gsignal()`

Description of Error:

Cause and Disposition:

`msg alloc failure in psignal()`

Description of Error:

Cause and Disposition:

`msg send failure in gsignal()`

Description of Error:

Cause and Disposition:

`msg send failure in psignal()`

Description of Error:

Cause and Disposition:

`surprise breakpoint`

Description of Error:

Cause and Disposition:

10.6 CONVEX UNIX/VIOP Panic Messages

The following messages are displayed by CONVEX UNIX when it encounters a problem in the VIOP with which it cannot cope.

`ccuopen: can't get windows`

Description of Error:

Cause and Disposition:

`ccurw bcount`

Description of Error:

Cause and Disposition:

`ccurw no ccu_win`

Description of Error:

Cause and Disposition:

`ddattach: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`ddattach: pq_fm_free error`

Description of Error:

Cause and Disposition:

`ddclose: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`ddclose: pq_fm_free error`

Description of Error:

Cause and Disposition:

`ddclose: tm_send error`

Description of Error:

Cause and Disposition:

`ddopen: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`ddopen: pq_fm_free error`

Description of Error:

Cause and Disposition:

`ddopen: tm_send error`

Description of Error:

Cause and Disposition:

`ddprobe: pq_fm_alloc error`

Description of Error:

Cause and Disposition:

`ddprobe: pq_fm_free error`

Description of Error:

Cause and Disposition:

`msg alloc failure in jp_command()`

Description of Error:

Cause and Disposition:

`msg send failure in jp_command()`

Description of Error:

Cause and Disposition:

`no bq recv bufs available???`

Description of Error:

Cause and Disposition:

`surprise breakpoint`

Description of Error:

Cause and Disposition:

`ve_postrecvbufs: not enough mbufs for initialization`

Description of Error:

Cause and Disposition:

`veinit: no bq message receive buffers`

Description of Error:

Cause and Disposition:

`veinit: not enough mbufs for initialization`

Description of Error:

Cause and Disposition:

`ve_receive_msg contains NULL mbuf chain pointer`

Description of Error:

Cause and Disposition:

Appendix A
Crash Dump Procedure

A.1 Crash Dump Procedure

When any C200 machine hangs or crashes, a crash dump is required for analysis of the problem. A crash dump contains the state of the machine at the moment the error occurred and allows personnel at CONVEX to determine cause of the problem. The following procedure should be followed to obtain a crash dump:

1. Determine the cause of the problem:
   a. Turn on the console printer
   b. Type a ^p at the (spu)> prompt (^p = CTRL p — obtains SPU prompt)
   c. Enter **cd /mnt/os** (change to operating system directory)
   d. Enter **syshang** (determine if software or hardware problem)
2. If the problem is software:
   a. Mount a 1/2-inch tape on magnetic tape unit 0
   b. Enter **cpureg**
   c. Enter **crashdump** (this may take a long time, depending on memory size)
   d. Boot the machine (go to step 4)
3. If the problem is hardware:
   a. Mount a 1/2-inch tape on magnetic tape unit 0
   b. Enter **cd /hw/cputest**
   c. Enter **hwdump** [*fn*] (creates two files named [fn] and [fn].rings)
   d. Boot the machine (go to step 4)
4. Clean up and boot the machine:
   a. Enter **osclean** (kill all active processes)
   b. Enter **sysreset** (reset the system)
   c. Enter **mminit** (initializes memory—repeat until no errors are encountered)
   d. Enter **boot_cpu single.cmd** (boot to single user)
   e. Enter **preen** (repeat until no errors are encountered)
   f. Enter **^d** (go to multi-user mode)
   g. Send the tape to CONVEX (go to step 6)
5. If problem was hardware (step 3 was executed):
   a. Enter **spu -r /hw/cputest/[fn] > [fn]** (transfer [fn] to system disk)
   b. Enter **spu -r /hw/cputest/[fn].rings > [fn].rings**
   c. Enter **tar cv <fn> <fn>.rings** (write files to tape)
6. Send the information (tape and printouts) to the attention of the Product Support Specialist assisting you at the following address:

Technical Assistance Center
CONVEX COMPUTER CORP
701 North Plano Road
Richardson, Texas 75081

Appendix B
CPU Instruction Glossary

B.1 Overview

When a diagnostic test fails, it will generally indicate the "instruction under test" at the moment that the test failed. It is important to note, however, that the instruction under test is not always the instruction that caused the test to fail.

The following table lists instruction abbreviations, a description of each, and which board is most likely to have the problem. If multiple boards are listed, they appear in decending order of probability.

B.2 CPU Instruction Glossary

Instruction Mnemonic	Instruction Description	Suspect Board(s)
add.b Sj,Sk	Add scalar/scalar integer byte	ASP
add.b Vi,Sj,Vk	Add vector/scalar integer byte	VPC, VPD, ASP
add.b Vi,Vj,Vk	Add vector/vector integer byte	VPC, VPD
add.b.f Vi,Sj,Vk	Add vector/scalar byte using VM	VPC, VPD, ASP
add.b.f Vi,Vj,Vk	Add vector/vector byte using not VM	VPC, VPD
add.b.t Vi,Sj,Vk	Add vector/scalar byte using VM	VPC, VPD, ASP
add.b.t Vi,Vj,Vk	Add vector/vector byte using VM	VPC, VPD
add.d Sj,Sk	Add scalar/scalar double float	SFU, ASP, IPP
add.d Vi,Sj,Vk	Add vector/scalar double float	VPC, VPD, ASP
add.d Vi,Vj,Vk	Add vector/vector double float	VPC, VPD
add.d.f Vi,Sj,Vk	Add vector/scalar double using not VM	VPC, VPD, ASP
add.d.f Vi,Vj,Vk	Add vector/vector double using not VM	VPC, VPD
add.d.t Vi,Sj,Vk	Add vector/scalar double using VM	VPC, VPD, ASP
add.d.t Vi,Vj,Vk	Add vector/vector double using VM	VPC, VPD
add.h #N,Ak	Add immediate address halfword	ASP, IPP
add.h #N,Sk	Add scalar/immediate integer halfword	ASP, IPP
add.h #n,Ak	Add short immediate address halfword	ASP, IPP
add.h Aj,Ak	Add address register halfword	ASP
add.h Sj,Sk	Add scalar/scalar integer halfword	ASP
add.h Vi,Sj,Vk	Add vector/scalar integer halfword	VPC, VPD, ASP
add.h Vi,Vj,Vk	Add vector/vector integer halfword	VPC, VPD
add.h.f Vi,Sj,Vk	Add vector/scalar halfword using not VM	VPC, VPD, ASP
add.h.f Vi,Vj,Vk	Add vector/vector halfword using not VM	VPC, VPD
add.h.t Vi,Sj,Vk	Add vector/scalar halfword using VM	VPC, VPD, ASP
add.h.t Vi,Vj,Vk	Add vector/vector halfword using VM	VPC, VPD

add.l Sj,Sk	Add scalar/scalar integer longword	ASP
add.l Vi,Sj,Vk	Add vector/scalar integer longword	VPC, VPD, ASP
add.l Vi,Vj,Vk	Add vector/vector integer longword	VPC, VPD
add.l.f Vi,Sj,Vk	Add vector/scalar longword using not VM	VPC, VPD, ASP
add.l.f Vi,Vj,Vk	Add vector/vector longword using not VM	VPC, VPD
add.l.t Vi,Sj,Vk	Add vector/scalar longword using VM	VPC, VPD, ASP
add.l.t Vi,Vj,Vk	Add vector/vector longword using VM	VPC, VPD
add.s #N,Sk	Add scalar/immediate single float	SFU, ASP, IPP
add.s Sj,Sk	Add scalar/scalar single float	SFU, ASP, IPP
add.s Vi,Sj,Vk	Add vector/scalar single float	VPC, VPD, ASP
add.s Vi,Vj,Vk	Add vector/vector single float	VPC, VPD
add.s.f Vi,Sj,Vk	Add vector/scalar single using not VM	VPC, VPD, ASP
add.s.f Vi,Vj,Vk	Add vector/vector single using not VM	VPC, VPD
add.s.t Vi,Sj,Vk	Add vector/scalar single using VM	VPC, VPD, ASP
add.s.t Vi,Vj,Vk	Add vector/vector single using VM	VPC, VPD
add.w #N,Ak	Add immediate address word	ASP, IPP
add.w #N,Sk	Add scalar/immediate integer word	ASP, IPP
add.w #n,Ak	Add short immediate address word	ASP, IPP
add.w Aj,Ak	Add address register word	ASP
add.w Sj,Ak	Add scalar to address word	ASP
add.w Sj,Sk	Add scalar/scalar integer word	ASP
add.w Vi,Sj,Vk	Add vector/scalar integer word	VPC, VPD, ASP
add.w Vi,Vj,Vk	Add vector/vector integer word	VPC, VPD
add.w.f Vi,Sj,Vk	Add vector/scalar word using not VM	VPC, VPD, ASP
add.w.f Vi,Vj,Vk	Add vector/vector word using not VM	VPC, VPD
add.w.t Vi,Sj,Vk	Add vector/scalar word using VM	VPC, VPD, ASP
add.w.t Vi,Vj,Vk	Add vector/vector word using VM	VPC, VPD
all Vk	AND reduce a vector	VPC, VPD, ASP
all.f Vk	AND reduce a vector using not VM	VPC, VPD, ASP
all.t Vk	AND reduce a vector using VM	VPC, VPD, ASP
and #N,Ak	AND immediate to address register	ASP, IPP
and #N,Sk	AND scalar/immediate	ASP, IPP
and Aj,Ak	AND address register	ASP
and Sj,Sk	AND scalar/scalar	ASP
and Vi,Sj,Vk	AND vector/scalar	VPC, VPD, ASP
and Vi,Vj,Vk	AND two vectors	VPC, VPD
and.f Vi,Sj,Vk	AND vector/scalar using not VM	VPC, VPD, ASP
and.f Vi,Vj,Vk	AND two vectors using not VM	VPC, VPD
and.t Vi,Sj,Vk	AND vector/scalar using VM	VPC, VPD, ASP
and.t Vi,Vj,Vk	AND two vectors using VM	VPC, VPD
any Vk	OR reduce a vector	VPC, VPD, ASP
any.f Vk	OR reduce a vector using not VM	VPC, VPD, ASP
any.t Vk	OR reduce a vector using VM	VPC, VPD, ASP
atan.d Sk	Arc-tangent of a double precision number	SFU, ASP, IPP
atan.s Sk	Arc-tangent of a single precision number	SFU, ASP, IPP

bkpt	Breakpoint	ASP, DCU, SFU, MCM
br	Branch always	IPP, ASP
bra.f	Branch on address carry false	IPP, ASP
bra.t	Branch on address carry true	IPP, ASP
bri.f	Branch on ION false	IPP, ASP
bri.t	Branch on ION true	IPP, ASP
brs.f	Branch on scalar carry false	IPP, ASP
brs.t	Branch on scalar carry true	IPP, ASP
call <effa>	Call a subroutine, long frame	ASP, DCU, SFU, MCM
callq <effa>	Push the PC and jump	ASP, DCU, SFU, MCM
calls <effa>	Call a subroutine, short frame	ASP, DCU, SFU, MCM
cfork	Clear a fork event	ASP, CPX, IPP
cos.d Sk	Cosine of a double precision number	SFU, ASP, IPP
cos.s Sk	Cosine of a single precision number	SFU, ASP, IPP
cprs.f Vj,Vk	Compress a vector using not VM	VPC, VPD
cprs.t Vj,Vk	Compress a vector using VM	VPC, VPD
ctrsg	Global synchronize CPU timers	ASP, CPX, IPP
cvtb.w Aj,Ak	Convert byte to word	ASP
cvtb.w Sj,Sk	Convert byte to word	ASP
cvtb.w Vj,Vk	Convert byte to word	VPC, VPD
cvtb.w.f Vj,Vk	Convert byte to word using not VM	VPC, VPD
cvtb.w.t Vj,Vk	Convert byte to word using VM	VPC, VPD
cvtd.l Sj,Sk	Convert double float to longword	SFU, ASP, IPP
cvtd.l Vj,Vk	Convert double float to longword	VPC, VPD
cvtd.l.f Vj,Vk	Convert double to longword using not VM	VPC, VPD
cvtd.l.t Vj,Vk	Convert double to longword using VM	VPC, VPD
cvtd.s Sj,Sk	Convert double float to single float	SFU, ASP, IPP
cvtd.s Vj,Vk	Convert double float to single float	VPC, VPD
cvtd.s.f Vj,Vk	Convert double to single using not VM	VPC, VPD
cvtd.s.t Vj,Vk	Convert double to single using VM	VPC, VPD
cvtd.w Sj,Sk	Convert double float to word	SFU, ASP, IPP
cvtd.w Vj,Vk	Convert double to word	VPC, VPD
cvtd.w.f Vj,Vk	Convert double to word using not VM	VPC, VPD
cvtd.w.t Vj,Vk	Convert double to word using VM	VPC, VPD
cvth.w Aj,Ak	Convert half to word	ASP
cvth.w Sj,Sk	Convert halfword to word	ASP
cvth.w Vj,Vk	Convert halfword to word	VPC, VPD
cvth.w.f Vj,Vk	Convert halfword to word using not VM	VPC, VPD
cvth.w.t Vj,Vk	Convert halfword to word using VM	VPC, VPD
cvtl.d Sj,Sk	Convert longword to double float	SFU, ASP, IPP
cvtl.d Vj,Vk	Convert longword to double float	VPC, VPD
cvtl.d.f Vj,Vk	Convert longword to double using not VM	VPC, VPD
cvtl.d.t Vj,Vk	Convert longword to double using VM	VPC, VPD
cvtl.s Sj,Sk	Convert longword to single float	SFU, ASP, IPP
cvtl.s Vj,Vk	Convert longword to single float	VPC, VPD

cvtl.s.f Vj,Vk	Convert longword to single using not VM	VPC, VPD
cvtl.s.t Vj,Vk	Convert longword to single using VM	VPC, VPD
cvtl.w Sj,Sk	Convert longword to word	ASP
cvtl.w Vj,Vk	Convert longword to word	VPC, VPD
cvtl.w.f Vj,Vk	Convert longword to word using not VM	VPC, VPD
cvtl.w.t Vj,Vk	Convert longword to word using VM	VPC, VPD
cvts.d Sj,Sk	Convert single float to double float	SFU, ASP, IPP
cvts.d Vj,Vk	Convert single float to double float	VPC, VPD
cvts.d.f Vj,Vk	Convert single to double using not VM	VPC, VPD
cvts.d.t Vj,Vk	Convert single to double using VM	VPC, VPD
cvts.l Sj,Sk	Convert single float to longword	SFU, ASP, IPP
cvts.l Vj,Vk	Convert single float to longword	VPC, VPD
cvts.l.f Vj,Vk	Convert single to longword using not VM	VPC, VPD
cvts.l.t Vj,Vk	Convert single to longword using VM	VPC, VPD
cvts.w Sj,Sk	Convert single float to word	SFU, ASP, IPP
cvts.w Vj,Vk	Convert single float to word	VPC, VPD
cvts.w.f Vj,Vk	Convert single to word using not VM	VPC, VPD
cvts.w.t Vj,Vk	Convert single to word using VM	VPC, VPD
cvtw.b Aj,Ak	Convert word to byte	ASP
cvtw.b Sj,Sk	Convert word to byte	ASP
cvtw.b Vj,Vk	Convert word to byte	VPC, VPD
cvtw.b.f Vj,Vk	Convert word to byte using not VM	VPC, VPD
cvtw.b.t Vj,Vk	Convert word to byte using VM	VPC, VPD
cvtw.d Sj,Sk	Convert word to double float	SFU, ASP, IPP
cvtw.d Vj,Vk	Convert word to double	VPC, VPD
cvtw.d.f Vj,Vk	Convert word to double using not VM	VPC, VPD
cvtw.d.t Vj,Vk	Convert word to double using VM	VPC, VPD
cvtw.h Aj,Ak	Convert word to halfword	ASP
cvtw.h Sj,Sk	Convert word to halfword	ASP
cvtw.h Vj,Vk	Convert word to halfword	VPC, VPD
cvtw.h.f Vj,Vk	Convert word to halfword using not VM	VPC, VPD
cvtw.h.t Vj,Vk	Convert word to halfword using VM	VPC, VPD
cvtw.l Sj,Sk	Convert word to longword	ASP
cvtw.l Vj,Vk	Convert word to longword	VPC, VPD
cvtw.l.f Vj,Vk	Convert word to longword using not VM	VPC, VPD
cvtw.l.t Vj,Vk	Convert word to longword using VM	VPC, VPD
cvtw.s Sj,Sk	Convert word to single float	SFU, ASP, IPP
cvtw.s Vj,Vk	Convert word to single float	VPC, VPD
cvtw.s.f Vj,Vk	Convert word to single using not VM	VPC, VPD
cvtw.s.t Vj,Vk	Convert word to single using VM	VPC, VPD
diag Ak	Execute nonstandard microcode sequence	ASP
div.b Sj,Sk	Divide scalar/scalar integer byte	SFU, ASP, IPP
div.b Vi,Sj,Vk	Divide vector/scalar integer byte	VPC, VPD, ASP
div.b Vi,Vj,Vk	Divide vector/vector integer byte	VPC, VPD
div.b.f Vi,Sj,Vk	Divide vector/scalar byte using not VM	VPC, VPD, ASP

div.b.f Vi,Vj,Vk	Divide byte vectors using not VM	VPC, VPD
div.b.t Vi,Sj,Vk	Divide vector/scalar byte using VM	VPC, VPD, ASP
div.b.t Vi,Vj,Vk	Divide byte vectors using VM	VPC, VPD
div.d Si,Vj,Vk	Divide scalar/vector double float	VPC, VPD, ASP
div.d Sj,Sk	Divide scalar/scalar double float	SFU, ASP, IPP
div.d Vi,Sj,Vk	Divide vector/scalar double float	VPC, VPD, ASP
div.d Vi,Vj,Vk	Divide vector/vector double float	VPC, VPD
div.d.f Si,Vj,Vk	Divide scalar/vector double using not VM	VPC, VPD, ASP
div.d.f Vi,Sj,Vk	Divide vector/scalar double using not VM	VPC, VPD, ASP
div.d.f Vi,Vj,Vk	Divide double vectors using not VM	VPC, VPD
div.d.t Si,Vj,Vk	Divide scalar/vector double using VM	VPC, VPD, ASP
div.d.t Vi,Sj,Vk	Divide vector/scalar double using VM	VPC, VPD, ASP
div.d.t Vi,Vj,Vk	Divide double vectors using VM	VPC, VPD
div.h #N,Ak	Divide immediate address halfword	SFU, ASP, IPP
div.h #N,Sk	Divide scalar/scalar integer halfword	SFU, ASP, IPP
div.h #n,Ak	Divide short immediate address halfword	SFU, ASP, IPP
div.h Aj,Ak	Divide address register halfword	SFU, ASP, IPP
div.h Sj,Sk	Divide scalar/scalar integer halfword	SFU, ASP, IPP
div.h Vi,Sj,Vk	Divide vector/scalar integer halfword	VPC, VPD, ASP
div.h Vi,Vj,Vk	Divide vector/vector integer halfword	VPC, VPD
div.h.f Vi,Sj,Vk	Divide vector/scalar halfword using not VM	VPC, VPD, ASP
div.h.f Vi,Vj,Vk	Divide halfword vectors using not VM	VPC, VPD
div.h.t Vi,Sj,Vk	Divide vector/scalar halfword using VM	VPC, VPD, ASP
div.h.t Vi,Vj,Vk	Divide halfword vectors using VM	VPC, VPD
div.l Sj,Sk	Divide scalar/scalar integer longword	SFU, ASP, IPP
div.l Vi,Sj,Vk	Divide vector/scalar integer longword	VPC, VPD, ASP
div.l Vi,Vj,Vk	Divide vector/vector integer longword	VPC, VPD
div.l.f Vi,Sj,Vk	Divide vector/scalar longword using not VM	VPC, VPD, ASP
div.l.f Vi,Vj,Vk	Divide longword vectors using not VM	VPC, VPD
div.l.t Vi,Sj,Vk	Divide vector/scalar longword using VM	VPC, VPD, ASP
div.l.t Vi,Vj,Vk	Divide longword vectors using VM	VPC, VPD
div.s #N,Sk	Divide scalar/scalar single float	SFU, ASP, IPP
div.s Si,Vj,Vk	Divide scalar/vector single float	VPC, VPD, ASP
div.s Sj,Sk	Divide scalar/scalar single float	SFU, ASP, IPP
div.s Vi,Sj,Vk	Divide vector/scalar single float	VPC, VPD, ASP
div.s Vi,Vj,Vk	Divide vector/vector single float	VPC, VPD
div.s.f Si,Vj,Vk	Divide scalar/vector single using not VM	VPC, VPD, ASP
div.s.f Vi,Sj,Vk	Divide vector/scalar single using not VM	VPC, VPD, ASP
div.s.f Vi,Vj,Vk	Divide single vectors using not VM	VPC, VPD
div.s.t Si,Vj,Vk	Divide scalar/vector single using VM	VPC, VPD, ASP
div.s.t Vi,Sj,Vk	Divide vector/scalar single using VM	VPC, VPD, ASP
div.s.t Vi,Vj,Vk	Divide single vectors using VM	VPC, VPD
div.w #N,Ak	Divide immediate address word	SFU, ASP, IPP
div.w #N,Sk	Divide scalar/scalar integer word	SFU, ASP, IPP
div.w #n,Ak	Divide short immediate address word	SFU, ASP, IPP

div.w Aj,Ak	Divide address register word	SFU, ASP, IPP
div.w Sj,Sk	Divide scalar/scalar integer word	SFU, ASP, IPP
div.w Vi,Sj,Vk	Divide vector/scalar integer word	VPC, VPD, ASP
div.w Vi,Vj,Vk	Divide vector/vector integer word	VPC, VPD
div.w.f Vi,Sj,Vk	Divide vector/scalar word using not VM	VPC, VPD, ASP
div.w.f Vi,Vj,Vk	Divide word vectors using not VM	VPC, VPD
div.w.t Vi,Sj,Vk	Divide vector/scalar word using VM	VPC, VPD, ASP
div.w.t Vi,Vj,Vk	Divide word vectors using VM	VPC, VPD
dsi	Disable interrupts; reset ION to 0	ASP, CPX, IPP
enag Sj,Sk	Enable all global CPU interrupts	ASP, CPX, IPP
enal Sj,Sk	Enable local CPU interrupt	ASP, CPX, IPP
eni	Enable interrupts; set ION to 1	ASP, CPX, IPP
eq.b Sj,Sk	Compare equal byte	ASP
eq.b Sj,Vk	Compare equal byte	VPC, VPD, ASP
eq.b Vj,Vk	Compare equal byte	VPC, VPD
eq.b.f Sj,Vk	Cmp. equal byte using not VM	VPC, VPD, ASP
eq.b.f Vj,Vk	Cmp. equal byte using not VM	VPC, VPD
eq.b.t Sj,Vk	Cmp. equal byte using VM	VPC, VPD, ASP
eq.b.t Vj,Vk	Cmp. equal byte using VM	VPC, VPD
eq.d Sj,Sk	Compare equal double float	SFU, ASP, IPP
eq.d Sj,Vk	Compare equal double precision	VPC, VPD, ASP
eq.d Vj,Vk	Compare equal double precision	VPC, VPD
eq.d.f Sj,Vk	Cmp. equal double using not VM	VPC, VPD, ASP
eq.d.f Vj,Vk	Cmp. equal double using not VM	VPC, VPD
eq.d.t Sj,Vk	Cmp. equal double using VM	VPC, VPD, ASP
eq.d.t Vj,Vk	Cmp. equal double using VM	VPC, VPD
eq.h #N,Ak	Compare equal halfword	ASP, IPP
eq.h #N,Sk	Compare equal halfword	ASP, IPP
eq.h #n,Ak	Compare equal halfword	ASP, IPP
eq.h Aj,Ak	Compare equal halfword	ASP
eq.h Sj,Sk	Compare equal halfword	ASP
eq.h Sj,Vk	Compare equal halfword	VPC, VPD, ASP
eq.h Vj,Vk	Compare equal halfword	VPC, VPD
eq.h.f Sj,Vk	Cmp. equal halfword using not VM	VPC, VPD, ASP
eq.h.f Vj,Vk	Cmp. equal halfword using not VM	VPC, VPD
eq.h.t Sj,Vk	Cmp. equal halfword using VM	VPC, VPD, ASP
eq.h.t Vj,Vk	Cmp. equal halfword using VM	VPC, VPD
eq.l Sj,Sk	Compare equal longword	ASP
eq.l Sj,Vk	Compare equal longword	VPC, VPD, ASP
eq.l Vj,Vk	Compare equal longword	VPC, VPD
eq.l.f Sj,Vk	Cmp. equal long using not VM	VPC, VPD, ASP
eq.l.f Vj,Vk	Cmp. equal long using not VM	VPC, VPD
eq.l.t Sj,Vk	Cmp. equal long using VM	VPC, VPD, ASP
eq.l.t Vj,Vk	Cmp. equal long using VM	VPC, VPD
eq.s Sj,Sk	Compare equal single float	SFU, ASP, IPP

eq.s Sj,Vk	Compare equal single	VPC, VPD, ASP
eq.s Vj,Vk	Compare equal single	VPC, VPD
eq.s.f Sj,Vk	Cmp. equal single using not VM	VPC, VPD, ASP
eq.s.f Vj,Vk	Cmp. equal single using not VM	VPC, VPD
eq.s.t Sj,Vk	Cmp. equal single using VM	VPC, VPD, ASP
eq.s.t Vj,Vk	Cmp. equal single using VM	VPC, VPD
eq.w #N,Ak	Compare equal word	ASP, IPP
eq.w #N,Sk	Compare equal word	ASP, IPP
eq.w #n,Ak	Compare equal word	ASP, IPP
eq.w Aj,Ak	Compare equal word	ASP
eq.w Sj,Sk	Compare equal word	ASP
eq.w Sj,Vk	Compare equal word	VPC, VPD, ASP
eq.w Vj,Vk	Compare equal word	VPC, VPD
eq.w.f Sj,Vk	Cmp. equal word using not VM	VPC, VPD, ASP
eq.w.f Vj,Vk	Cmp. equal word using not VM	VPC, VPD
eq.w.t Sj,Vk	Cmp. equal word using VM	VPC, VPD, ASP
eq.w.t Vj,Vk	Cmp. equal word using VM	VPC, VPD
exit	Error exit instruction	DCU, SFU, ASP, MCM
exp.d Sk	Exponent of a double precision number	SFU, ASP, IPP
exp.s Sk	Exponent of a single precision number	SFU, ASP, IPP
frint.d Sj,Sk	Integerize float double scalar	SFU, ASP, IPP
frint.d Vj,Vk	Integerize float double vector	VPC, VPD
frint.d.f Vj,Vk	Integerize double vector using not VM	VPC, VPD
frint.d.t Vj,Vk	Integerize double vector using VM	VPC, VPD
frint.s Sj,Sk	Integerize float single scalar	SFU, ASP, IPP
frint.s Vj,Vk	Integerize float single vector	VPC, VPD
frint.s.f Vj,Vk	Integerize single vector using not VM	VPC, VPD
frint.s.t Vj,Vk	Integerize single vector using VM	VPC, VPD
get.l <Ceffa>,Sk	Get communication/scalar	ASP, CPX, IPP
get.w <Ceffa>,Ak	Get communication/address	ASP, CPX, IPP
halt #N,Ak	Halt the CPU	ASP, IPP
idle	Idle the CPU	ASP, CPX, IPP
inc.l <Ceffa>,Sk	Increment communication/scalar	ASP, CPX, IPP
inc.w <Ceffa>,Ak	Increment communication/address	ASP, CPX, IPP
incr.l <effa>,Sk	Increment long resource structure	DCU, SFU, ASP, MCM
incr.w <effa>,Ak	Increment resource structure data	DCU, SFU, ASP, MCM
jmp <effa>	Jump always	IPP, ASP
jmpa.f <effa>	Jump on address carry false	IPP, ASP
jmpa.t <effa>	Jump on address carry true	IPP, ASP
jmpi.f <effa>	Jump on ION false	IPP, ASP
jmpi.t <effa>	Jump on ION true	IPP, ASP
jmps.f <effa>	Jump on scalar carry false	IPP, ASP
jmps.t <effa>	Jump on scalar carry true	IPP, ASP
join	Join all threads	ASP, CPX, IPP
lck <Ceffa>	Lock communication register	ASP, CPX, IPP

ld.b <effa>,Ak	Load address register byte	DCU, SFU, ASP, MCM
ld.b <effa>,Sk	Load scalar byte	DCU, SFU, ASP, MCM
ld.b <effa>,Vk	Load vector byte	VPC, VPD, DCU, SFU, ASP, MCM
ld.b.f <effa>,Vk	Load vector byte using not VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.b.t <effa>,Vk	Load vector byte using VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.d #N,Sk	Load immediate upper 32 bits	ASP, IPP
ld.d <effa>,Sk	Load scalar double float	DCU, SFU, ASP, MCM
ld.d <effa>,Vk	Load vector double float	VPC, VPD, DCU, SFU, ASP, MCM
ld.d.f <effa>,Vk	Load vector double float using not VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.d.t <effa>,Vk	Load vector double float using VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.dl #N,Sk	Load 64-bit floating immediate, lower half	ASP, IPP
ld.du #N,Sk	Load 64-bit floating immediate, upper half	ASP, IPP
ld.h #N,Ak	Load halfword immediate into Ak	ASP, IPP
ld.h #n,Ak	Load short immediate into Ak	ASP, IPP
ld.h <effa>,Ak	Load address register halfword	DCU, SFU, ASP, MCM
ld.h <effa>,Sk	Load scalar halfword	DCU, SFU, ASP, MCM
ld.h <effa>,Vk	Load vector halfword	VPC, VPD, DCU, SFU, ASP, MCM
ld.h.f <effa>,Vk	Load vector halfword using not VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.h.t <effa>,Vk	Load vector halfword using VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.l #N,Sk	Load 32-bit immediate sign-extended to 64 bits	ASP, IPP
ld.l <effa>,Sk	Load scalar longword	DCU, SFU, ASP, MCM
ld.l <effa>,VLS	Load VS and VL from memory	ASP, DCU, SFU, VPC, MCM
ld.l <effa>,Vk	Load vector longword	VPC, VPD, DCU, SFU, ASP, MCM
ld.l.f <effa>,Vk	Load vector longword using not VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.l.t <effa>,Vk	Load vector longword using VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.ll #N,Sk	Load 64-bit integer immediate, lower half	ASP, IPP
ld.lu #N,Sk	Load 64-bit integer immediate, upper half	ASP, IPP
ld.s <effa>,Sk	Load scalar single float	DCU, SFU, ASP, MCM
ld.s <effa>,Vk	Load vector single float	VPC, VPD, DCU, SFU, ASP, MCM
ld.s.f <effa>,Vk	Load vector single float using not VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.s.t <effa>,Vk	Load vector single float using VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.u #N,Sk	Load immediate, upper half	ASP, IPP
ld.w #N,Ak	Load immediate into Ak	ASP, IPP
ld.w #N,Sk	Load a 32-bit immediate	ASP, IPP
ld.w #N,VL	Load VL with an immediate	ASP, VPC, SFU, IPP
ld.w #N,VS	Load VS from an immediate	ASP, SFU, IPP
ld.w #n,Ak	Load short immediate into Ak	ASP, IPP
ld.w <effa>,Ak	Load address register word	DCU, SFU, ASP, MCM
ld.w <effa>,Sk	Load scalar word	DCU, SFU, ASP, MCM
ld.w <effa>,Vk	Load vector word	VPC, VPD, DCU, SFU, ASP, MCM
ld.w.f <effa>,Vk	Load vector word using not VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.w.t <effa>,Vk	Load vector word using VM	VPC, VPD, DCU, SFU, ASP, MCM
ld.x <effa>, VM	Load VM from memory	VPC, DCU, SFU, ASP, MCM
ldcmr <effa>,Ak	Load communication registers	DCU, SFU, ASP, MCM
ldea <effa>,Ak	Load effective address	ASP

ldea <effa>,Sk	Load effective address/scalar	ASP
ldkdr Ak	Load all eight SDRs	DCU, SFU, ASP, MCM
ldpa Aj,Ak	Load a physical byte address into Ak	DCU, SFU, ASP, MCM
ldsdr Ak	Load process SDRs	DCU, SFU, ASP, MCM
ldvi.b Vj,Vk	Index load vector byte	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.b.f Vj,Vk	Index Load vector byte using not VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.b.t Vj,Vk	Index Load vector byte using VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.d Vj,Vk	Index load vector double float	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.d.f Vj,Vk	Index Load vector double using not VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.d.t Vj,Vk	Index Load vector double using VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.h Vj,Vk	Index load vector halfword	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.h.f Vj,Vk	Index Load vector halfword using not VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.h.t Vj,Vk	Index Load vector halfword using VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.l Vj,Vk	Index load vector longword	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.l.f Vj,Vk	Index Load vector longword using not VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.l.t Vj,Vk	Index Load vector longword using VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.s Vj,Vk	Index load vector single float	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.s.f Vj,Vk	Index Load vector single using not VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.s.t Vj,Vk	Index Load vector single using VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.w Vj,Vk	Index load vector word	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.w.f Vj,Vk	Index Load vector word using not VM	VPC, VPD, ASP, DCU, SFU, MCM
ldvi.w.t Vj,Vk	Index Load vector word using VM	VPC, VPD, ASP, DCU, SFU, MCM
le.b Sj,Sk	Compare less than or equal byte	ASP
le.b Sj,Vk	Compare less than or equal byte	VPC, VPD, ASP
le.b Vj,Vk	Compare less than or equal byte	VPC, VPD
le.b.f Sj,Vk	Cmp. less than or equal byte using not VM	VPC, VPD, ASP
le.b.f Vj,Vk	Cmp. less than or equal byte using not VM	VPC, VPD
le.b.t Sj,Vk	Cmp. less than or equal byte using VM	VPC, VPD, ASP
le.b.t Vj,Vk	Cmp. less than or equal byte using VM	VPC, VPD
le.d Sj,Sk	Compare less than or equal double float	SFU, ASP, IPP
le.d Sj,Vk	Compare less than or equal double float	VPC, VPD, ASP
le.d Vj,Vk	Compare less than or equal double float	VPC, VPD
le.d.f Sj,Vk	Cmp. less than or equal double using not VM	VPC, VPD, ASP
le.d.f Vj,Vk	Cmp. less than or equal double using not VM	VPC, VPD
le.d.t Sj,Vk	Cmp. less than or equal double using VM	VPC, VPD, ASP
le.d.t Vj,Vk	Cmp. less than or equal double using VM	VPC, VPD
le.h #N,Ak	Compare less than or equal halfword	ASP, IPP
le.h #N,Sk	Compare less than or equal halfword	ASP, IPP
le.h #n,Ak	Compare less than or equal halfword	ASP, IPP
le.h Aj,Ak	Compare less than or equal signed halfword	ASP
le.h Sj,Sk	Compare less than or equal halfword	ASP
le.h Sj,Vk	Compare less than or equal halfword	VPC, VPD, ASP
le.h Vj,Vk	Compare less than or equal halfword	VPC, VPD
le.h.f Sj,Vk	Cmp. less than or equal half using not VM	VPC, VPD, ASP
le.h.f Vj,Vk	Cmp. less than or equal half using not VM	VPC, VPD

le.h.t Sj,Vk	Cmp. less than or equal half using VM	VPC, VPD, ASP
le.h.t Vj,Vk	Cmp. less than or equal half using VM	VPC, VPD
le.l Sj,Sk	Compare less than or equal longword	ASP
le.l Sj,Vk	Compare less than or equal longword	VPC, VPD, ASP
le.l Vj,Vk	Compare less than or equal longword	VPC, VPD
le.l.f Sj,Vk	Cmp. less than or equal long using not VM	VPC, VPD, ASP
le.l.f Vj,Vk	Cmp. less than or equal long using not VM	VPC, VPD
le.l.t Sj,Vk	Cmp. less than or equal long using VM	VPC, VPD, ASP
le.l.t Vj,Vk	Cmp. less than or equal long using VM	VPC, VPD
le.s #N,Sk	Compare less than or equal single	SFU, ASP, IPP
le.s Sj,Sk	Compare less than or equal single float	SFU, ASP, IPP
le.s Sj,Vk	Compare less than or equal single	VPC, VPD, ASP
le.s Vj,Vk	Compare less than or equal single	VPC, VPD
le.s.f Sj,Vk	Cmp. less than or equal single using not VM	VPC, VPD, ASP
le.s.f Vj,Vk	Cmp. less than or equal single using not VM	VPC, VPD
le.s.t Sj,Vk	Cmp. less than or equal single using VM	VPC, VPD, ASP
le.s.t Vj,Vk	Cmp. less than or equal single using VM	VPC, VPD
le.w #N,Ak	Compare less than or equal word	ASP, IPP
le.w #N,Sk	Compare less than or equal word	ASP, IPP
le.w #n,Ak	Compare less than or equal word	ASP, IPP
le.w Aj,Ak	Compare less than or equal signed word	ASP
le.w Sj,Sk	Compare less than or equal word	ASP
le.w Sj,Vk	Compare less than or equal word	VPC, VPD, ASP
le.w Vj,Vk	Compare less than or equal word	VPC, VPD
le.w.f Sj,Vk	Cmp. less than or equal word using not VM	VPC, VPD, ASP
le.w.f Vj,Vk	Cmp. less than or equal word using not VM	VPC, VPD
le.w.t Sj,Vk	Cmp. less than or equal word using VM	VPC, VPD, ASP
le.w.t Vj,Vk	Cmp. less than or equal word using VM	VPC, VPD
leu.b Sj,Sk	Compare less than or equal to byte	ASP
leu.h #N,Ak	Compare unsigned less than halfword	ASP, IPP
leu.h #N,Sk	Compare unsigned less than or equal to halfword	ASP, IPP
leu.h #n,Ak	Compare unsigned less than or equal halfword	ASP, IPP
leu.h Aj,Ak	Compare unsigned less than or equal to halfword	ASP
leu.h Sj,Sk	Compare less than or equal to halfword	ASP
leu.l Sj,Sk	Compare less than or equal to longword	ASP
leu.w #N,Ak	Compare unsigned less than word	ASP, IPP
leu.w #N,Sk	Compare unsigned less than or equal to word	ASP, IPP
leu.w #n,Ak	Compare unsigned less than or equal word	ASP, IPP
leu.w Aj,Ak	Compare unsigned less than or equal to word	ASP
leu.w Sj,Sk	Compare less than or equal to word	ASP
ln.d Sk	Natural logarithm of a double precision number	SFU, ASP, IPP
ln.s Sk	Natural logarithm of a single precision number	SFU, ASP, IPP
lop Sj,Sk	Count of leading zeros in Sj	SFU, ASP, IPP
lop Vj,Vk	Leading ones position vector	VPC, VPD
lop.f Vj,Vk	Leading ones position vector using not VM	VPC, VPD

lop.t Vj,Vk	Leading ones position vector using VM	VPC, VPD
lt.b Sj,Sk	Compare less than byte	ASP
lt.b Sj,Vk	Compare less than byte	VPC, VPD, ASP
lt.b Vj,Vk	Compare less than byte	VPC, VPD
lt.b.f Sj,Vk	Cmp. less than byte using not VM	VPC, VPD, ASP
lt.b.f Vj,Vk	Cmp. less than byte using not VM	VPC, VPD
lt.b.t Sj,Vk	Cmp. less than byte using VM	VPC, VPD, ASP
lt.b.t Vj,Vk	Cmp. less than byte using VM	VPC, VPD
lt.d Sj,Sk	Compare less than double float	SFU, ASP, IPP
lt.d Sj,Vk	Compare less than double float	VPC, VPD, ASP
lt.d Vj,Vk	Compare less than double float	VPC, VPD
lt.d.f Sj,Vk	Cmp. less than double using not VM	VPC, VPD, ASP
lt.d.f Vj,Vk	Cmp. less than double using not VM	VPC, VPD
lt.d.t Sj,Vk	Cmp. less than double using VM	VPC, VPD, ASP
lt.d.t Vj,Vk	Cmp. less than double using VM	VPC, VPD
lt.h #N,Ak	Compare less than halfword	ASP, IPP
lt.h #N,Sk	Compare less than halfword	ASP, IPP
lt.h #n,Ak	Compare less than halfword	ASP, IPP
lt.h Aj,Ak	Compare less than signed halfword	ASP
lt.h Sj,Sk	Compare less than halfword	ASP
lt.h Sj,Vk	Compare less than halfword	VPC, VPD, ASP
lt.h Vj,Vk	Compare less than halfword	VPC, VPD
lt.h.f Sj,Vk	Cmp. less than halfword using not VM	VPC, VPD, ASP
lt.h.f Vj,Vk	Cmp. less than halfword using not VM	VPC, VPD
lt.h.t Sj,Vk	Cmp. less than halfword using VM	VPC, VPD, ASP
lt.h.t Vj,Vk	Cmp. less than halfword using VM	VPC, VPD
lt.l Sj,Sk	Compare less than longword	ASP
lt.l Sj,Vk	Compare less than longword	VPC, VPD, ASP
lt.l Vj,Vk	Compare less than longword	VPC, VPD
lt.l.f Sj,Vk	Cmp. less than long using not VM	VPC, VPD, ASP
lt.l.f Vj,Vk	Cmp. less than long using not VM	VPC, VPD
lt.l.t Sj,Vk	Cmp. less than long using VM	VPC, VPD, ASP
lt.l.t Vj,Vk	Cmp. less than long using VM	VPC, VPD
lt.s #N,Sk	Compare less than single	SFU, ASP, IPP
lt.s Sj,Sk	Compare less than single float	SFU, ASP, IPP
lt.s Sj,Vk	Compare less than single	VPC, VPD, ASP
lt.s Vj,Vk	Compare less than single	VPC, VPD
lt.s.f Sj,Vk	Cmp. less than single using not VM	VPC, VPD, ASP
lt.s.f Vj,Vk	Cmp. less than single using not VM	VPC, VPD
lt.s.t Sj,Vk	Cmp. less than single using VM	VPC, VPD, ASP
lt.s.t Vj,Vk	Cmp. less than single using VM	VPC, VPD
lt.w #N,Ak	Compare less than word	ASP, IPP
lt.w #N,Sk	Compare less than word	ASP, IPP
lt.w #n,Ak	Compare less than word	ASP, IPP
lt.w Aj,Ak	Compare less than signed word	ASP

lt.w Sj,Sk	Compare less than word	ASP
lt.w Sj,Vk	Compare less than word	VPC, VPD, ASP
lt.w Vj,Vk	Compare less than word	VPC, VPD
lt.w.f Sj,Vk	Cmp. less than word using not VM	VPC, VPD, ASP
lt.w.f Vj,Vk	Cmp. less than word using not VM	VPC, VPD
lt.w.t Sj,Vk	Cmp. less than word using VM	VPC, VPD, ASP
lt.w.t Vj,Vk	Cmp. less than word using VM	VPC, VPD
ltu.b Sj,Sk	Compare less than byte	ASP
ltu.h #N,Ak	Compare unsigned less than halfword	ASP, IPP
ltu.h #N,Sk	Compare unsigned less than halfword	ASP, IPP
ltu.h #n,Ak	Compare unsigned less than halfword	ASP, IPP
ltu.h Aj,Ak	Compare unsigned less than halfword	ASP
ltu.h Sj,Sk	Compare less than halfword	ASP
ltu.l Sj,Sk	Compare less than longword	ASP
ltu.w #N,Ak	Compare unsigned less than word	ASP, IPP
ltu.w #N,Sk	Compare unsigned less than word	ASP, IPP
ltu.w #n,Ak	Compare unsigned less than word	ASP, IPP
ltu.w Aj,Ak	Compare unsigned less than word	ASP
ltu.w Sj,Sk	Compare less than word	ASP
mask.f Vi,Sj,Vk	Mask vector/scalar using not VM	VPC, VPD, ASP
mask.t Vi,Sj,Vk	Mask vector/scalar using VM	VPC, VPD, ASP
mask.t Vi,Vj,Vk	Mask vector/vector	VPC, VPD
mat.l Sk,<Ceffa>	Match scalar/communication	ASP, CPX, IPP
mat.w Ak,<Ceffa>	Match address/communication	ASP, CPX, IPP
matm.l Sk,<effa>	Match scalar register and memory	DCU, SFU, ASP, MCM
matm.w Ak,<effa>	Match address register and memory	DCU, SFU, ASP, MCM
max.b Vk	Max of a vector of bytes	VPC, VPD, ASP
max.b.f Vk	Max of vector of bytes using not VM	VPC, VPD, ASP
max.b.t Vk	Max of vector of bytes using VM	VPC, VPD, ASP
max.d Vk	Max of a vector of double float	VPC, VPD, ASP
max.d.f Vk	Max of vector of doubles using not VM	VPC, VPD, ASP
max.d.t Vk	Max of vector of doubles using VM	VPC, VPD, ASP
max.h Vk	Max of a vector of halfwords	VPC, VPD, ASP
max.h.f Vk	Max of vector of halfwords using not VM	VPC, VPD, ASP
max.h.t Vk	Max of vector of halfwords using VM	VPC, VPD, ASP
max.l Vk	Max of a vector of longwords	VPC, VPD, ASP
max.l.f Vk	Max of vector of longwords using not VM	VPC, VPD, ASP
max.l.t Vk	Max of vector of longwords using VM	VPC, VPD, ASP
max.s Vk	Max of a vector of single float	VPC, VPD, ASP
max.s.f Vk	Max of vector of singles using not VM	VPC, VPD, ASP
max.s.t Vk	Max of vector of singles using VM	VPC, VPD, ASP
max.w Vk	Max of a vector of words	VPC, VPD, ASP
max.w.f Vk	Max of vector of words using not VM	VPC, VPD, ASP
max.w.t Vk	Max of vector of words using VM	VPC, VPD, ASP
merg.f Vi,Sj,Vk	Merge vector/scalar using not VM	VPC, VPD, ASP

merg.t Vi,Sj,Vk	Merge vector/scalar	VPC, VPD, ASP
merg.t Vi,Vj,Vk	Merge vector/vector	VPC, VPD
min.b Vk	Min of a vector of bytes	VPC, VPD, ASP
min.b.f Vk	Min of vector of bytes using not VM	VPC, VPD, ASP
min.b.t Vk	Min of vector of bytes using VM	VPC, VPD, ASP
min.d Vk	Min of a vector of double float	VPC, VPD, ASP
min.d.f Vk	Min of vector of doubles using not VM	VPC, VPD, ASP
min.d.t Vk	Min of vector of doubles using VM	VPC, VPD, ASP
min.h Vk	Min of a vector of halfwords	VPC, VPD, ASP
min.h.f Vk	Min of vector of halfwords using not VM	VPC, VPD, ASP
min.h.t Vk	Min of vector of halfwords using VM	VPC, VPD, ASP
min.l Vk	Min of a vector of longwords	VPC, VPD, ASP
min.l.f Vk	Min of vector of longwords using not VM	VPC, VPD, ASP
min.l.t Vk	Min of vector of longwords using VM	VPC, VPD, ASP
min.s Vk	Min of a vector of single float	VPC, VPD, ASP
min.s.f Vk	Min of vector of singles using not VM	VPC, VPD, ASP
min.s.t Vk	Min of vector of singles using VM	VPC, VPD, ASP
min.w Vk	Min of a vector of words	VPC, VPD, ASP
min.w.f Vk	Min of vector of words using not VM	VPC, VPD, ASP
min.w.t Vk	Min of vector of words using VM	VPC, VPD, ASP
mov Aj,Ak	Move address register	ASP
mov Aj,Sk	Move an address to a scalar	ASP
mov Ak,PSW	Load an address register into the PSW	ASP, SFU, VPC
mov Ak,VL	Move Ak to VL	ASP, VPC, SFU
mov Ak,VS	Move Ak to VS	ASP, SFU
mov CIR,Sk	Move Comm Index Register to a scalar	ASP
mov CPUID,Sk	Move CPU identification to Scalar	ASP
mov ICR,Sk	Move Interrupt Control Register to Scalar	ASP, CPX, IPP
mov ITR,Sk	Move the ITC, ITSR, NITC into Sk	ASP, CPX, IPP
mov PC,Ak	Load next PC address	ASP, IPP
mov PSW,Ak	Store the PSW into an address register	ASP, SFU, VPC
mov Si,Sj,Vk	Move a scalar to a vector element	VPC, VPD, ASP
mov Sj,Ak	Move 32 bits of Sj into Ak	ASP
mov Sj,Sk,VM	Load VM(Sj) from Sk	VPC, ASP
mov Sj,VM,Sk	Load Sk from VM(Sj)	VPC, VPD, ASP
mov Sk,CIR	Move scalar to Comm Index Register	ASP, DCU
mov Sk,ICR	Move Scalar to Interrupt Control Register	ASP, CPX, IPP
mov Sk,ITR	Load NITC, ITC, ITSR from Sk	ASP, CPX, IPP
mov Sk,ITSR	Load ITSR with a scalar	ASP, CPX, IPP
mov Sk,TCPU	Move Scalar to Target CPU Register	ASP, CPX, IPP
mov Sk,TID	Load Thread ID from Scalar	ASP, DCU
mov Sk,TTR	Move scalar to thread timer	ASP, CPX, IPP
mov Sk,VML	Load VM<63..0> from Sk	VPC, ASP
mov Sk,VMU	Load VM<127..64> from Sk	VPC, ASP
mov Sk,VV	Move scalar to vector valid flag	ASP, IPP

mov TCPU,Sk	Move the Target CPU Register to Scalar	ASP, CPX, IPP
mov TID,Sk	Load scalar with Thread ID	ASP
mov TOC,Sk	Move TOC to a scalar	ASP, CPX, IPP
mov TTR,Sk	Move thread timer/scalar	ASP, CPX, IPP
mov VL,Ak	Move VL to Ak	ASP
mov VML,Sk	Load Sk from VM<63..0>	VPC, VPD, ASP
mov VMU,Sk	Load Sk from VM<127..64>	VPC, VPD, ASP
mov VS,Ak	Move VS to Ak	ASP
mov Vi,Sj,Sk	Move a vector element to a scalar	VPC, VPD, ASP
mov.d Sj,Sk	Move scalar register single float	ASP
mov.l Sj,Sk	Move scalar register longword	ASP
mov.s Sj,Sk	Move scalar register double float	ASP
mov.w Sj,Sk	Move scalar register word	ASP
mov.w Sk,VL	Move Sk to VL	ASP, VPC, SFU
mov.w Sk,VS	Move Sk to VS	ASP, SFU
mov.w VL,Sk	Move VL to Sk	ASP
mov.w VS,Sk	Move VS to Sk	ASP
mski Sk	Mask out interrupt	ASP, CPX, IPP
msync	Synchronize stores to memory	ASP, DCU, MCM
mul.b Sj,Sk	Multiply scalar/scalar integer byte	SFU, ASP, IPP
mul.b Vi,Sj,Vk	Multiply vector/scalar integer byte	VPC, VPD, ASP
mul.b Vi,Vj,Vk	Multiply vector/vector integer byte	VPC, VPD
mul.b.f Vi,Sj,Vk	Multiply vector/scalar byte using not VM	VPC, VPD, ASP
mul.b.f Vi,Vj,Vk	Multiply byte vectors using not VM	VPC, VPD
mul.b.t Vi,Sj,Vk	Multiply vector/scalar byte using VM	VPC, VPD, ASP
mul.b.t Vi,Vj,Vk	Multiply byte vectors using VM	VPC, VPD
mul.d Sj,Sk	Multiply scalar/scalar double float	SFU, ASP, IPP
mul.d Vi,Sj,Vk	Multiply vector/scalar double float	VPC, VPD, ASP
mul.d Vi,Vj,Vk	Multiply vector/vector double float	VPC, VPD
mul.d.f Vi,Sj,Vk	Multiply vector/scalar double using not VM	VPC, VPD, ASP
mul.d.f Vi,Vj,Vk	Multiply double vectors using not VM	VPC, VPD
mul.d.t Vi,Sj,Vk	Multiply vector/scalar double using VM	VPC, VPD, ASP
mul.d.t Vi,Vj,Vk	Multiply double vectors using VM	VPC, VPD
mul.h #N,Ak	Multiply immediate address halfword	SFU, ASP, IPP
mul.h #N,Sk	Multiply scalar/immediate integer halfword	SFU, ASP, IPP
mul.h #n,Ak	Multiply short immediate address halfword	SFU, ASP, IPP
mul.h Aj,Ak	Multiply address register halfword	SFU, ASP, IPP
mul.h Sj,Sk	Multiply scalar/scalar integer halfword	SFU, ASP, IPP
mul.h Vi,Sj,Vk	Multiply vector/scalar integer halfword	VPC, VPD, ASP
mul.h Vi,Vj,Vk	Multiply vector/vector integer halfword	VPC, VPD
mul.h.f Vi,Sj,Vk	Multiply vector/scalar halfword using not VM	VPC, VPD, ASP
mul.h.f Vi,Vj,Vk	Multiply halfword vectors using not VM	VPC, VPD
mul.h.t Vi,Sj,Vk	Multiply vector/scalar halfword using VM	VPC, VPD, ASP
mul.h.t Vi,Vj,Vk	Multiply halfword vectors using VM	VPC, VPD
mul.l Sj,Sk	Multiply scalar/scalar integer longword	SFU, ASP, IPP

mul.l Vi,Sj,Vk	Multiply vector/scalar integer longword	VPC, VPD, ASP
mul.l Vi,Vj,Vk	Multiply vector/vector integer longword	VPC, VPD
mul.l.f Vi,Sj,Vk	Multiply vector/scalar longword using not VM	VPC, VPD, ASP
mul.l.f Vi,Vj,Vk	Multiply longword vectors using not VM	VPC, VPD
mul.l.t Vi,Sj,Vk	Multiply vector/scalar longword using VM	VPC, VPD, ASP
mul.l.t Vi,Vj,Vk	Multiply longword vectors using VM	VPC, VPD
mul.s #N,Sk	Multiply scalar/immediate single float	SFU, ASP, IPP
mul.s Sj,Sk	Multiply scalar/scalar single float	SFU, ASP, IPP
mul.s Vi,Sj,Vk	Multiply vector/scalar single float	VPC, VPD, ASP
mul.s Vi,Vj,Vk	Multiply vector/vector single float	VPC, VPD
mul.s.f Vi,Sj,Vk	Multiply vector/scalar single using not VM	VPC, VPD, ASP
mul.s.f Vi,Vj,Vk	Multiply single vectors using not VM	VPC, VPD
mul.s.t Vi,Sj,Vk	Multiply vector/scalar single using VM	VPC, VPD, ASP
mul.s.t Vi,Vj,Vk	Multiply single vectors using VM	VPC, VPD
mul.w #N,Ak	Multiply immediate address word	SFU, ASP, IPP
mul.w #N,Sk	Multiply scalar/immediate integer word	SFU, ASP, IPP
mul.w #n,Ak	Multiply short immediate address word	SFU, ASP, IPP
mul.w Aj,Ak	Multiply address register word	SFU, ASP, IPP
mul.w Sj,Sk	Multiply scalar/scalar integer word	SFU, ASP, IPP
mul.w Vi,Sj,Vk	Multiply vector/scalar integer word	VPC, VPD, ASP
mul.w Vi,Vj,Vk	Multiply vector/vector integer word	VPC, VPD
mul.w.f Vi,Sj,Vk	Multiply vector/scalar word using not VM	VPC, VPD, ASP
mul.w.f Vi,Vj,Vk	Multiply word vectors using not VM	VPC, VPD
mul.w.t Vi,Sj,Vk	Multiply vector/scalar word using VM	VPC, VPD, ASP
mul.w.t Vi,Vj,Vk	Multiply word vectors using VM	VPC, VPD
neg.b Sj,Sk	Negate scalar/scalar integer byte	ASP
neg.b Vj,Vk	Negate vector/vector integer byte	VPC, VPD
neg.b.f Vj,Vk	Negate byte vector using not VM	VPC, VPD
neg.b.t Vj,Vk	Negate byte vector using VM	VPC, VPD
neg.d Sj,Sk	Negate scalar/scalar double float	SFU, ASP, IPP
neg.d Vj,Vk	Negate vector/vector double float	VPC, VPD
neg.d.f Vj,Vk	Negate double using not VM	VPC, VPD
neg.d.t Vj,Vk	Negate double using VM	VPC, VPD
neg.h Aj,Ak	Negate address register halfword	ASP
neg.h Sj,Sk	Negate scalar/scalar integer halfword	ASP
neg.h Vj,Vk	Negate vector/vector integer halfword	VPC, VPD
neg.h.f Vj,Vk	Negate halfword using not VM	VPC, VPD
neg.h.t Vj,Vk	Negate halfword using VM	VPC, VPD
neg.l Sj,Sk	Negate scalar/scalar integer longword	ASP
neg.l Vj,Vk	Negate vector/vector integer longword	VPC, VPD
neg.l.f Vj,Vk	Negate longword using not VM	VPC, VPD
neg.l.t Vj,Vk	Negate longword using VM	VPC, VPD
neg.s Sj,Sk	Negate scalar/scalar single float	SFU, ASP, IPP
neg.s Vj,Vk	Negate vector/vector single float	VPC, VPD
neg.s.f Vj,Vk	Negate single using not VM	VPC, VPD

neg.s.t Vj,Vk	Negate single using VM	VPC, VPD
neg.w Aj,Ak	Negate address register word	ASP
neg.w Sj,Sk	Negate scalar/scalar integer word	ASP
neg.w Vj,Vk	Negate vector/vector integer word	VPC, VPD
neg.w.f Vj,Vk	Negate word using not VM	VPC, VPD
neg.w.t Vj,Vk	Negate word using VM	VPC, VPD
nop	No operation (branch never)	IPP
not Aj,Ak	Complement address register	ASP
not Sj,Sk	Complement scalar/scalar	ASP
not Vj,Vk	Complement a vector	VPC, VPD
not.f Vj,Vk	Complement a vector using not VM	VPC, VPD
not.t Vj,Vk	Complement a vector using VM	VPC, VPD
or #N,Ak	OR immediate to address register	ASP, IPP
or #N,Sk	OR scalar/immediate	ASP, IPP
or Aj,Ak	OR address register	ASP
or Sj,Sk	OR scalar/scalar	ASP
or Vi,Sj,Vk	OR vector/scalar	VPC, VPD, ASP
or Vi,Vj,Vk	OR two vectors	VPC, VPD
or.f Vi,Sj,Vk	OR vector/scalar using not VM	VPC, VPD, ASP
or.f Vi,Vj,Vk	OR two vectors using not VM	VPC, VPD
or.t Vi,Sj,Vk	OR vector/scalar using VM	VPC, VPD, ASP
or.t Vi,Vj,Vk	OR two vectors using VM	VPC, VPD
parity Vk	Exclusive OR reduce a vector	VPC, VPD, ASP
parity.f Vk	Exclusive OR reduce vector using not VM	VPC, VPD, ASP
parity.t Vk	Exclusive OR reduce vector using VM	VPC, VPD, ASP
pate Ak	Purge ATU entry	DCU, ASP, IPP, SFU
patu	Purge the entire ATU	DCU, ASP
pbkpt	Force process breakpoint exception	DCU, SFU, ASP, MCM
pfork <effa>,Ak	Post a fork event	ASP, CPX, IPP
pich	Purge the Icache	ASP, IPP
plc.f VM,Sk	Load the number of 0's in VM into Sk	VPC, VPD, SFU, ASP
plc.t Sj,Sk	Count the number of 1's in Sj	SFU, ASP, IPP
plc.t VM,Sk	Load the number of 1's in VM into Sk	VPC, VPD, SFU, ASP
plc.t Vj,Vk	Population count of a vector	VPC, VPD
plc.t.f Vj,Vk	Population count of vector using not VM	VPC, VPD
plc.t.t Vj,Vk	Population count of vector using VM	VPC, VPD
plch	Purge the Lcache	ASP
pop.l Sk	Pop Sk <63..0> from the stack	DCU, SFU, ASP, MCM
pop.w Ak	Pop word into address register	DCU, SFU, ASP, MCM
pop.w Sk	Pop Sk <31..0> from the stack	DCU, SFU, ASP, MCM
popr Ak,<effa>	Pop resource/address register	DCU, SFU, ASP, MCM
prod.b Vk	Multiply reduce a vector of bytes	VPC, VPD, ASP, SFU, IPP
prod.b.f Vk	Multiply reduce byte vector using not VM	VPC, VPD, ASP, SFU, IPP
prod.b.t Vk	Multiply reduce byte vector using VM	VPC, VPD, ASP, SFU, IPP
prod.d Vk	Multiply reduce a vector of double float	VPC, VPD, ASP, SFU, IPP

prod.d.f Vk	Multiply reduce double vector using not VM	VPC, VPD, ASP, SFU, IPP
prod.d.t Vk	Multiply reduce double vector using VM	VPC, VPD, ASP, SFU, IPP
prod.h Vk	Multiply reduce a vector of halfwords	VPC, VPD, ASP, SFU, IPP
prod.h.f Vk	Multiply reduce halfword vector using not VM	VPC, VPD, ASP, SFU, IPP
prod.h.t Vk	Multiply reduce halfword vector using VM	VPC, VPD, ASP, SFU, IPP
prod.l Vk	Multiply reduce a vector of longwords	VPC, VPD, ASP, SFU, IPP
prod.l.f Vk	Multiply reduce longword vector using not VM	VPC, VPD, ASP, SFU, IPP
prod.l.t Vk	Multiply reduce longword vector using VM	VPC, VPD, ASP, SFU, IPP
prod.s Vk	Multiply reduce a vector of single float	VPC, VPD, ASP, SFU, IPP
prod.s.f Vk	Multiply reduce single vector using not VM	VPC, VPD, ASP, SFU, IPP
prod.s.t Vk	Multiply reduce single vector using VM	VPC, VPD, ASP, SFU, IPP
prod.w Vk	Multiply reduce a vector of words	VPC, VPD, ASP, SFU, IPP
prod.w.f Vk	Multiply reduce word vector using not VM	VPC, VPD, ASP, SFU, IPP
prod.w.t Vk	Multiply reduce word vector using VM	VPC, VPD, ASP, SFU, IPP
psh.l Sk	Push Sk<63..0> onto the stack	DCU, ASP, MCM
psh.w Ak	Push an address register	DCU, ASP, MCM
psh.w Sk	Push Sk<31..0> onto the stack	DCU, ASP, MCM
pshea <effa>	Push effective address	DCU, ASP, MCM
pshr Ak,<effa>	Push address register/resource	DCU, SFU, ASP, MCM
put.l Sk,<Ceffa>	Put scalar/communication	ASP, CPX, IPP
put.w Ak,<Ceffa>	Put address/communication	ASP, CPX, IPP
rcv.l <Ceffa>,Sk	Receive communication/scalar	ASP, CPX, IPP
rcv.w <Ceffa>,Ak	Receive communication/address	ASP, CPX, IPP
rcvr.l <effa>,Sk	Receive scalar register/resource	DCU, SFU, ASP, MCM
rcvr.w <effa>,Ak	Receive address register/resource	DCU, SFU, ASP, MCM
rtn	Return from subroutine call	DCU, SFU, ASP, MCM
rtnc	Return from a context block	DCU, SFU, ASP, MCM
rtni	Return from base level interrupt	DCU, SFU, ASP, MCM
rtnq	Pop the PC and jump	DCU, SFU, ASP, MCM
shf #N,Ak	Logical shift immediate to address register	SFU, ASP, IPP
shf #N,Sk	Shift scalar/immediate	SFU, ASP, IPP
shf #n,Ak	Logical shift short immediate	ASP, IPP
shf Aj,Ak	Shift an address	SFU, ASP, IPP
shf Sj,Sk	Shift a scalar	SFU, ASP, IPP
shf Sj,Vk	Shift a vector accumulator	VPC, VPD, ASP
shf Vi,Sj,Vk	Shift a vector accumulator	VPC, VPD, ASP
shf Vi,Vj,Vk	Shift vector/vector	VPC, VPD
shf.f Sj,Vk	Shift vector/scalar using not VM	VPC, VPD, ASP
shf.f Vi,Sj,Vk	Shift vector/scalar using not VM	VPC, VPD, ASP
shf.f Vi,Vj,Vk	Shift vector/vector using not VM	VPC, VPD
shf.t Sj,Vk	Shift vector/scalar using VM	VPC, VPD, ASP
shf.t Vi,Sj,Vk	Shift vector/scalar using VM	VPC, VPD, ASP
shf.t Vi,Vj,Vk	Shift vector/vector using VM	VPC, VPD
shf.w #N,Sk	Shift Scalar word/immediate	SFU, ASP, IPP
shf.w Sj,Sk	Shift a scalar word	SFU, ASP, IPP

sin.d Sk	Sine of a double precision number	SFU, ASP, IPP
sin.s Sk	Sine of a single precision number	SFU, ASP, IPP
snd.l Sk,<Ceffa>	Send scalar/communication	ASP, CPX, IPP
snd.w Ak,<Ceffa>	Send address/communication	ASP, CPX, IPP
sndr.l Sk,<effa>	Send scalar register/resource	DCU, SFU, ASP, MCM
sndr.w Ak,<effa>	Send address register/resource	DCU, SFU, ASP, MCM
spawn <effa>,Ak	Spawn a fork event	ASP, CPX, IPP
sqrt.d Sk	Square root of a double precision number	SFU, ASP, IPP
sqrt.d Vj,Vk	Square root double vector/vector	VPC, VPD
sqrt.d.f Vj,Vk	Square root double using not VM	VPC, VPD
sqrt.d.t Vj,Vk	Square root double using VM	VPC, VPD
sqrt.s Sk	Square root of a single precision number	SFU, ASP, IPP
sqrt.s Vj,Vk	Square root single vector/vector	VPC, VPD
sqrt.s.f Vj,Vk	Square root single using not VM	VPC, VPD
sqrt.s.t Vj,Vk	Square root single using VM	VPC, VPD
st.b Ak,<effa>	Store address register byte	DCU, ASP, MCM
st.b Sk,<effa>	Store scalar byte	DCU, ASP, MCM
st.b Vk,<effa>	Store vector byte	VPC, VPD, DCU, SFU, ASP, MCM
st.b.f Vk,<effa>	Store vector byte using not VM	VPC, VPD, DCU, SFU, ASP, MCM
st.b.t Vk,<effa>	Store vector byte using VM	VPC, VPD, DCU, SFU, ASP, MCM
st.d Sk,<effa>	Store scalar double float	DCU, ASP, MCM
st.d Vk,<effa>	Store vector double float	VPC, VPD, DCU, SFU, ASP, MCM
st.d.f Vk,<effa>	Store vector double float using not VM	VPC, VPD, DCU, SFU, ASP, MCM
st.d.t Vk,<effa>	Store vector double float using VM	VPC, VPD, DCU, SFU, ASP, MCM
st.h Ak,<effa>	Store address register halfword	DCU, ASP, MCM
st.h Sk,<effa>	Store scalar halfword	DCU, ASP, MCM
st.h Vk,<effa>	Store vector halfword	VPC, VPD, DCU, SFU, ASP, MCM
st.h.f Vk,<effa>	Store vector halfword using not VM	VPC, VPD, DCU, SFU, ASP, MCM
st.h.t Vk,<effa>	Store vector halfword using VM	VPC, VPD, DCU, SFU, ASP, MCM
st.l Sk,<effa>	Store scalar longword	DCU, ASP, MCM
st.l VLS,<effa>	Store VS and VL to memory	VPC, VPD, DCU, SFU, ASP, MCM
st.l Vk,<effa>	Store vector longword	VPC, VPD, DCU, SFU, ASP, MCM
st.l.f Vk,<effa>	Store vector longword using not VM	VPC, VPD, DCU, SFU, ASP, MCM
st.l.t Vk,<effa>	Store vector longword using VM	VPC, VPD, DCU, SFU, ASP, MCM
st.s Sk,<effa>	Store scalar single float	DCU, ASP, MCM
st.s Vk,<effa>	Store vector single float	VPC, VPD, DCU, SFU, ASP, MCM
st.s.f Vk,<effa>	Store vector single float using not VM	VPC, VPD, DCU, SFU, ASP, MCM
st.s.t Vk,<effa>	Store vector single float using VM	VPC, VPD, DCU, SFU, ASP, MCM
st.w Ak,<effa>	Store address register word	DCU, ASP, MCM
st.w Sk,<effa>	Store scalar word	DCU, ASP, MCM
st.w Vk,<effa>	Store vector word	VPC, VPD, DCU, SFU, ASP, MCM
st.w.f Vk,<effa>	Store vector word using not VM	VPC, VPD, DCU, SFU, ASP, MCM
st.w.t Vk,<effa>	Store vector word using VM	VPC, VPD, DCU, SFU, ASP, MCM
st.x VM,<effa>	Store VM into memory	VPC, VPD, DCU, ASP, MCM
stcmr Ak,<effa>	Store communication registers	DCU, SFU, ASP, CPX, IPP, MCM

ste.b Sk,<effa>	Store an extended scalar byte	DCU, SFU, ASP, VPC, MCM
ste.b.f Sk,<effa	Store extended scalar byte using not VM	DCU, SFU, ASP, VPC, MCM
ste.b.t Sk,<effa>	Store extended scalar byte using VM	DCU, SFU, ASP, VPC, MCM
ste.d Sk,<effa>	Store an extended scalar double float	DCU, SFU, ASP, VPC, MCM
ste.d.f Sk,<effa>	Store extended scalar double using not VM	DCU, SFU, ASP, VPC, MCM
ste.d.t Sk,<effa>	Store extended scalar double using VM	DCU, SFU, ASP, VPC, MCM
ste.h Sk,<effa>	Store an extended scalar halfword	DCU, SFU, ASP, VPC, MCM
ste.h.f Sk,<effa>	Store extended scalar halfword using not VM	DCU, SFU, ASP, VPC, MCM
ste.h.t Sk,<effa>	Store extended scalar halfword using VM	DCU, SFU, ASP, VPC, MCM
ste.l Sk,<effa>	Store an extended scalar longword	DCU, SFU, ASP, VPC, MCM
ste.l.f Sk,<effa>	Store extended scalar longword using not VM	DCU, SFU, ASP, VPC, MCM
ste.l.t Sk,<effa>	Store extended scalar longword using VM	DCU, SFU, ASP, VPC, MCM
ste.s Sk,<effa>	Store an extended scalar single float	DCU, SFU, ASP, VPC, MCM
ste.s.f Sk,<effa>	Store extended scalar single using not VM	DCU, SFU, ASP, VPC, MCM
ste.s.t Sk,<effa>	Store extended scalar single using VM	DCU, SFU, ASP, VPC, MCM
ste.w Sk,<effa>	Store an extended scalar word	DCU, SFU, ASP, VPC, MCM
ste.w.f Sk,<effa>	Store extended scalar word using not VM	DCU, SFU, ASP, VPC, MCM
ste.w.t Sk,<effa>	Store extended scalar word using VM	DCU, SFU, ASP, VPC, MCM
stop	Stop CPU clocks	ASP
stvi.b Sk,Vj	Scalar index store vector byte	VPC, VPD, ASP, DCU, MCM
stvi.b Vk,Vj	Index store vector byte	VPC, VPD, ASP, DCU, MCM
stvi.b.f Sk,Vj	Scalar index store vector byte using not VM	VPC, VPD, ASP, DCU, MCM
stvi.b.f Vk,Vj	Index store vector byte using not VM	VPC, VPD, ASP, DCU, MCM
stvi.b.t Sk,Vj	Scalar index store vector byte using VM	VPC, VPD, ASP, DCU, MCM
stvi.b.t Vk,Vj	Index store vector byte using VM	VPC, VPD, ASP, DCU, MCM
stvi.d Sk,Vj	Scalar index store vector double float	VPC, VPD, ASP, DCU, MCM
stvi.d Vk,Vj	Index store vector double float	VPC, VPD, ASP, DCU, MCM
stvi.d.f Sk,Vj	Scalar index store vector double using not VM	VPC, VPD, ASP, DCU, MCM
stvi.d.f Vk,Vj	Index store vector double using not VM	VPC, VPD, ASP, DCU, MCM
stvi.d.t Sk,Vj	Scalar index store vector double using VM	VPC, VPD, ASP, DCU, MCM
stvi.d.t Vk,Vj	Index store vector double using VM	VPC, VPD, ASP, DCU, MCM
stvi.h Sk,Vj	Scalar index store vector halfword	VPC, VPD, ASP, DCU, MCM
stvi.h Vk,Vj	Index store vector halfword	VPC, VPD, ASP, DCU, MCM
stvi.h.f Sk,Vj	Scalar index store vector half using not VM	VPC, VPD, ASP, DCU, MCM
stvi.h.f Vk,Vj	Index store vector halfword using not VM	VPC, VPD, ASP, DCU, MCM
stvi.h.t Sk,Vj	Scalar index store vector half using VM	VPC, VPD, ASP, DCU, MCM
stvi.h.t Vk,Vj	Index store vector halfword using VM	VPC, VPD, ASP, DCU, MCM
stvi.l Sk,Vj	Scalar index store vector longword	VPC, VPD, ASP, DCU, MCM
stvi.l Vk,Vj	Index store vector longword	VPC, VPD, ASP, DCU, MCM
stvi.l.f Sk,Vj	Scalar index store vector long using not VM	VPC, VPD, ASP, DCU, MCM
stvi.l.f Vk,Vj	Index store vector longword using not VM	VPC, VPD, ASP, DCU, MCM
stvi.l.t Sk,Vj	Scalar index store vector long using VM	VPC, VPD, ASP, DCU, MCM
stvi.l.t Vk,Vj	Index store vector longword using VM	VPC, VPD, ASP, DCU, MCM
stvi.s Sk,Vj	Scalar index store vector single float	VPC, VPD, ASP, DCU, MCM
stvi.s Vk,Vj	Index store vector single float	VPC, VPD, ASP, DCU, MCM

stvi.s.f Sk,Vj	Scalar index store vector single using not VM	VPC, VPD, ASP, DCU, MCM
stvi.s.f Vk,Vj	Index store vector single using not VM	VPC, VPD, ASP, DCU, MCM
stvi.s.t Sk,Vj	Scalar index store vector single using VM	VPC, VPD, ASP, DCU, MCM
stvi.s.t Vk,Vj	Index store vector single using VM	VPC, VPD, ASP, DCU, MCM
stvi.w Sk,Vj	Scalar index store vector word	VPC, VPD, ASP, DCU, MCM
stvi.w Vk,Vj	Index store vector word	VPC, VPD, ASP, DCU, MCM
stvi.w.f Sk,Vj	Scalar index store vector word using not VM	VPC, VPD, ASP, DCU, MCM
stvi.w.f Vk,Vj	Index store vector word using not VM	VPC, VPD, ASP, DCU, MCM
stvi.w.t Sk,Vj	Scalar index store vector word using VM	VPC, VPD, ASP, DCU, MCM
stvi.w.t Vk,Vj	Index store vector word using VM	VPC, VPD, ASP, DCU, MCM
sub.b Sj,Sk	Subtract scalar/scalar integer byte	ASP
sub.b Vi,Sj,Vk	Subtract vector/scalar integer byte	VPC, VPD, ASP
sub.b Vi,Vj,Vk	Subtract vector/vector integer byte	VPC, VPD
sub.b.f Vi,Sj,Vk	Subtract vector/scalar byte using not VM	VPC, VPD, ASP
sub.b.f Vi,Vj,Vk	Subtract byte vectors using not VM	VPC, VPD
sub.b.t Vi,Sj,Vk	Subtract vector/scalar byte using VM	VPC, VPD, ASP
sub.b.t Vi,Vj,Vk	Subtract byte vectors using VM	VPC, VPD
sub.d Si,Vj,Vk	Subtract scalar/vector double float	VPC, VPD, ASP
sub.d Sj,Sk	Subtract scalar/scalar double float	SFU, ASP, IPP
sub.d Vi,Sj,Vk	Subtract vector/scalar double float	VPC, VPD, ASP
sub.d Vi,Vj,Vk	Subtract vector/vector double float	VPC, VPD
sub.d.f Si,Vj,Vk	Subtract scalar/vector double using not VM	VPC, VPD, ASP
sub.d.f Vi,Sj,Vk	Subtract vector/scalar double using not VM	VPC, VPD, ASP
sub.d.f Vi,Vj,Vk	Subtract double vectors using not VM	VPC, VPD
sub.d.t Si,Vj,Vk	Subtract scalar/vector double using VM	VPC, VPD, ASP
sub.d.t Vi,Sj,Vk	Subtract vector/scalar double using VM	VPC, VPD, ASP
sub.d.t Vi,Vj,Vk	Subtract double vectors using VM	VPC, VPD
sub.h #N,Ak	Subtract immediate address halfword	ASP, IPP
sub.h #N,Sk	Subtract scalar/immediate integer halfword	ASP, IPP
sub.h #n,Ak	Subtract short immediate address halfword	ASP, IPP
sub.h Aj,Ak	Subtract address register halfword	ASP
sub.h Sj,Sk	Subtract scalar/scalar integer halfword	ASP
sub.h Vi,Sj,Vk	Subtract vector/scalar integer halfword	VPC, VPD, ASP
sub.h Vi,Vj,Vk	Subtract vector/vector integer halfword	VPC, VPD
sub.h.f Vi,Sj,Vk	Subtract vector/scalar halfword using not VM	VPC, VPD, ASP
sub.h.f Vi,Vj,Vk	Subtract halfword vectors using not VM	VPC, VPD
sub.h.t Vi,Sj,Vk	Subtract vector/scalar halfword using VM	VPC, VPD, ASP
sub.h.t Vi,Vj,Vk	Subtract halfword vectors using VM	VPC, VPD
sub.l Sj,Sk	Subtract scalar/scalar integer longword	ASP
sub.l Vi,Sj,Vk	Subtract vector/scalar integer longword	VPC, VPD, ASP
sub.l Vi,Vj,Vk	Subtract vector/vector integer longword	VPC, VPD
sub.l.f Vi,Sj,Vk	Subtract vector/scalar longword using not VM	VPC, VPD, ASP
sub.l.f Vi,Vj,Vk	Subtract longword vectors using not VM	VPC, VPD
sub.l.t Vi,Sj,Vk	Subtract vector/scalar longword using VM	VPC, VPD, ASP
sub.l.t Vi,Vj,Vk	Subtract longword vectors using VM	VPC, VPD

sub.s #N,Sk	Subtract scalar/immediate single float	SFU, ASP, IPP
sub.s Si,Vj,Vk	Subtract scalar/vector single float	VPC, VPD, ASP
sub.s Sj,Sk	Subtract scalar/scalar single float	SFU, ASP, IPP
sub.s Vi,Sj,Vk	Subtract vector/scalar single float	VPC, VPD, ASP
sub.s Vi,Vj,Vk	Subtract vector/vector single float	VPC, VPD
sub.s.f Si,Vj,Vk	Subtract scalar/vector single using not VM	VPC, VPD, ASP
sub.s.f Vi,Sj,Vk	Subtract vector/scalar single using not VM	VPC, VPD, ASP
sub.s.f Vi,Vj,Vk	Subtract single vectors using not VM	VPC, VPD
sub.s.t Si,Vj,Vk	Subtract scalar/vector single using VM	VPC, VPD, ASP
sub.s.t Vi,Sj,Vk	Subtract vector/scalar single using VM	VPC, VPD, ASP
sub.s.t Vi,Vj,Vk	Subtract single vectors using VM	VPC, VPD
sub.w #N,Ak	Subtract immediate address word	ASP, IPP
sub.w #N,Sk	Subtract scalar/immediate integer word	ASP, IPP
sub.w #n,Ak	Subtract short immediate address word	ASP, IPP
sub.w Aj,Ak	Subtract address register word	ASP
sub.w Sj,Sk	Subtract scalar/scalar integer word	ASP
sub.w Vi,Sj,Vk	Subtract vector/scalar integer word	VPC, VPD, ASP
sub.w Vi,Vj,Vk	Subtract vector/vector integer word	VPC, VPD
sub.w.f Vi,Sj,Vk	Subtract vector/scalar word using not VM	VPC, VPD, ASP
sub.w.f Vi,Vj,Vk	Subtract word vectors using not VM	VPC, VPD
sub.w.t Vi,Sj,Vk	Subtract vector/scalar word using VM	VPC, VPD, ASP
sub.w.t Vi,Vj,Vk	Subtract word vectors using VM	VPC, VPD
sum.b Vk	Sum a vector of bytes	VPC, VPD, ASP
sum.b.f Vk	Sum a vector of bytes using not VM	VPC, VPD, ASP
sum.b.t Vk	Sum a vector of bytes using VM	VPC, VPD, ASP
sum.d Vk	Sum a vector of double float	VPC, VPD, ASP, SFU, IPP
sum.d.f Vk	Sum a vector of double using not VM	VPC, VPD, ASP, SFU, IPP
sum.d.t Vk	Sum a vector of double using VM	VPC, VPD, ASP, SFU, IPP
sum.h Vk	Sum a vector of halfwords	VPC, VPD, ASP
sum.h.f Vk	Sum a vector of halfwords using not VM	VPC, VPD, ASP
sum.h.t Vk	Sum a vector of halfwords using VM	VPC, VPD, ASP
sum.l Vk	Sum a vector of longwords	VPC, VPD, ASP
sum.l.f Vk	Sum a vector of longwords using not VM	VPC, VPD, ASP
sum.l.t Vk	Sum a vector of longwords using VM	VPC, VPD, ASP
sum.s Vk	Sum a vector of single float	VPC, VPD, ASP, SFU, IPP
sum.s.f Vk	Sum a vector of single using not VM	VPC, VPD, ASP, SFU, IPP
sum.s.t Vk	Sum a vector of single using VM	VPC, VPD, ASP, SFU, IPP
sum.w Vk	Sum a vector of words	VPC, VPD, ASP
sum.w.f Vk	Sum a vector of words using not VM	VPC, VPD, ASP
sum.w.t Vk	Sum a vector of words using VM	VPC, VPD, ASP
sysc #r,#g	Perform a system call	ASP, DCU, SFU, IPP
tac <effa>	Test and clear a byte in memory	DCU, SFU, ASP, MCM
tas <effa>	Test and set a memory byte	DCU, SFU, ASP, MCM
trap #rm,#b	Force a trap system exception	ASP, CPX, DCU, SFU, IPP
tst <Ceffa>	Test communication register lock bit	ASP, CPX

tstvv	Test value of vector valid flag	ASP
tzc Sj,Sk	Count of trailing zeros in Sj	SFU, ASP, IPP
tzc Vj,Vk	Trailing zero count vector	VPC, VPD
tzc.f Vj,Vk	Trailing zero count vector using not VM	VPC, VPD
tzc.t Vj,Vk	Trailing zero count vector using VM	VPC, VPD
ulk <Ceffa>	Unlock communication register	ASP, CPX, IPP
wfork	Wait for a fork event	ASP, CPX, IPP
xmti Sk	Transmit interrupt	ASP, PIA, SP2, CPX
xor #N,Ak	Exclusive OR immediate to address register	ASP, IPP
xor #N,Sk	Exclusive OR scalar/immediate	ASP, IPP
xor Aj,Ak	Exclusive OR address register	ASP
xor Sj,Sk	Exclusive OR scalar/scalar	ASP
xor Vi,Sj,Vk	Exclusive OR vector/scalar	VPC, VPD, ASP
xor Vi,Vj,Vk	Exclusive OR two vectors	VPC, VPD
xor.f Vi,Sj,Vk	Exclusive OR vector/scalar using not VM	VPC, VPD, ASP
xor.f Vi,Vj,Vk	Exclusive OR two vectors using not VM	VPC, VPD
xor.t Vi,Sj,Vk	Exclusive OR vector/scalar using VM	VPC, VPD, ASP
xor.t Vi,Vj,Vk	Exclusive OR two vectors using VM	VPC, VPD
xpnd.f Vj,Vk	Expand a vector using not VM	VPC, VPD
xpnd.t Vj,Vk	Expand a vector using VM	VPC, VPD

Appendix C
ASP Entry Point Listing

C.1 Overview

This appendix contains ASP entry point listings arranged (in tables) both by entry points and by opcodes.

Entry points are useful in troubleshooting a system hang or halt situation. The service processor *cpureg* command displays a dump of CPU registers. One of these registers (the **IPC** register) shows the value of the last instruction out of the dispatch latch of the instruction processor.

These tables show the correspondence of entry point values to instruction (opcodes) and can therefore be used to determine which instruction was dispatched immediately before the system hung. This may help identify the last thing the CPU attempted to do before the system hung.

Once the IPC register value has been obtained with the *cpureg* command, the value can be used to find the corresponding opcode in the Entry Point Listing. The instruction can then be looked up in the instruction glossary (Appendix B) to determine which boards are likely suspects for that particular opcode.

Finally, the *CONVEX Architecture Reference* can be consulted to determine what the opcode does and what areas of the system are involved in its operation.

> **NOTE**
>
> It is not always safe to assume that the instruction in the IPC register is the cause of the hang. Because the IPP is able to dispatch scalar and vector instructions asynchronously, it may dispatch a scalar instruction, then dispatch a vector instruction, and then hang on the *scalar* instruction. In this situation, the IPC register no longer contains the instruction (scalar) that caused the machine to hang. Instead, it contains the vector instruction, which just happened to be there when the system hung.
>
> In many cases, however, the instruction in the IPC register is the last instruction dispatched before the system hung *and* is also the cause of the hang.

C.2 ASP Entry Points — by Entry Points

Entry Point	Opcode	UPC
EP=000	EXIT	0C00H
EP=001	EXIT@	0C01H
EP=002	JMP	0C02H
EP=003	JMP@	0C03H
EP=004	JMPI.F	0C04H
EP=005	JMPI.F@	0C05H
EP=006	JMPI.T	0C06H
EP=007	JMPI.T@	0C07H
EP=008	JMPA.F	0C08H
EP=009	JMPA.F@	0C09H
EP=00a	JMPA.T	0C0AH
EP=00b	JMPA.T@	0C0BH
EP=00c	JMPS.F	0C0CH
EP=00d	JMPS.F@	0C0DH
EP=00e	JMPS.T	0C0EH
EP=00f	JMPS.T@	0C0FH
EP=010	TAC.B	0C10H
EP=011	TAC.B@	0C11H
EP=012	LDEA_A	0C12H
EP=013	LDEA_A@	0C13H
EP=014	LD.L_VLS	0C14H
EP=015	LD.L_VLS@	0C15H
EP=016	LD.X_VM	0C16H
EP=017	LD.X_VM@	0C17H
EP=018	TAS.B	0C18H
EP=019	TAS.B@	0C19H
EP=01a	PSHEA	0C1AH
EP=01b	PSHEA@	0C1BH
EP=01c	ST.L_VLS	0C1CH
EP=01d	ST.L_VLS@	0C1DH
EP=01e	ST.X_VM	0C1EH
EP=01f	ST.X_VM@	0C1FH
EP=020	HALT	0C20H
EP=021	SYSC	0C21H
EP=022	LD.H_IA	0C22H
EP=023	LD.W_IA	0C23H
EP=024	AND_IA	0C24H
EP=025	OR_IA	0C25H
EP=026	XOR_IA	0C26H
EP=027	SHF_IA	0C27H

EP=028	ADD.H_IA	0C28H
EP=029	ADD.W_IA	0C29H
EP=02a	SUB.H_IA	0C2AH
EP=02b	SUB.W_IA	0C2BH
EP=02c	MUL.H_IA	0C2CH
EP=02d	MUL.W_IA	0C2DH
EP=02e	DIV.H_IA	0C2EH
EP=02f	DIV.W_IA	0C2FH
EP=030	LD_IVL	0C30H
EP=031	LD_IVS	0C31H
EP=033	SHF.W_IS	0C33H
EP=034	TRAP	0C34H
EP=036	EQ.H_IA	0C36H
EP=037	EQ.W_IA	0C37H
EP=038	LEU.H_IA	0C38H
EP=039	LEU.W_IA	0C39H
EP=03a	LTU.H_IA	0C3AH
EP=03b	LTU.W_IA	0C3BH
EP=03c	LE.H_IA	0C3CH
EP=03d	LE.W_IA	0C3DH
EP=03e	LT.H_IA	0C3EH
EP=03f	LT.W_IA	0C3FH
EP=040	CALL	0C40H
EP=041	CALL@	0C41H
EP=042	CALLS	0C42H
EP=043	CALLS@	0C43H
EP=044	CALLQ	0C44H
EP=045	CALLQ@	0C45H
EP=046	PFORK	0C46H
EP=047	PFORK@	0C47H
EP=048	STE.B_S	0C48H
EP=049	STE.B_S@	0C49H
EP=04a	STE.H_S	0C4AH
EP=04b	STE.H_S@	0C4BH
EP=04c	STE.W_S	0C4CH
EP=04d	STE.W_S@	0C4DH
EP=04e	STE.L_S	0C4EH
EP=04f	STE.L_S@	0C4FH
EP=050	LD.B_A	0C50H
EP=051	LD.B_A@	0C51H
EP=052	LD.H_A	0C52H
EP=053	LD.H_A@	0C53H
EP=054	LD.W_A	0C54H
EP=055	LD.W_A@	0C55H
EP=056	INCR.W_A	0C56H

EP=057	INCR.W_A@	0C57H
EP=058	ST.B_A	0C58H
EP=059	ST.B_A@	0C59H
EP=05a	ST.H_A	0C5AH
EP=05b	ST.H_A@	0C5BH
EP=05c	ST.W_A	0C5CH
EP=05d	ST.W_A@	0C5DH
EP=05e	INCR.L_S	0C5EH
EP=05f	INCR.L_S@	0C5FH
EP=060	LD.B_S	0C60H
EP=061	LD.B_S@	0C61H
EP=062	LD.H_S	0C62H
EP=063	LD.H_S@	0C63H
EP=064	LD.W_S	0C64H
EP=065	LD.W_S@	0C65H
EP=066	LD.L_S	0C66H
EP=067	LD.L_S@	0C67H
EP=068	ST.B_S	0C68H
EP=069	ST.B_S@	0C69H
EP=06a	ST.H_S	0C6AH
EP=06b	ST.H_S@	0C6BH
EP=06c	ST.W_S	0C6CH
EP=06d	ST.W_S@	0C6DH
EP=06e	ST.L_S	0C6EH
EP=06f	ST.L_S@	0C6FH
EP=070	LD.B_V	0C70H
EP=071	LD.B_V@	0C71H
EP=072	LD.H_V	0C72H
EP=073	LD.H_V@	0C73H
EP=074	LD.W_V	0C74H
EP=075	LD.W_V@	0C75H
EP=076	LD.L_V	0C76H
EP=077	LD.L_V@	0C77H
EP=078	ST.B_V	0C78H
EP=079	ST.B_V@	0C79H
EP=07a	ST.H_V	0C7AH
EP=07b	ST.H_V@	0C7BH
EP=07c	ST.W_V	0C7CH
EP=07d	ST.W_V@	0C7DH
EP=07e	ST.L_V	0C7EH
EP=07f	ST.L_V@	0C7FH
EP=080	CVTW.B_A	0C80H
EP=081	CVTW.H_A	0C81H
EP=082	CVTB.W_A	0C82H
EP=083	CVTH.W_A	0C83H

EP=084	CVTW.B_S	0C84H
EP=085	CVTW.H_S	0C85H
EP=086	CVTB.W_S	0C86H
EP=087	CVTH.W_S	0C87H
EP=088	CVTW.S_S	0C88H
EP=089	CVTS.W_S	0C89H
EP=08a	CVTD.S_S	0C8AH
EP=08b	CVTS.D_S	0C8BH
EP=08c	CVTS.L_S	0C8CH
EP=08d	CVTD.L_S	0C8DH
EP=08e	CVTL.S_S	0C8EH
EP=08f	CVTL.D_S	0C8FH
EP=090	LDPA	0C90H
EP=091	SHF_JA	0C91H
EP=092	LD.H_JA	0C92H
EP=093	LD.W_JA	0C93H
EP=094	CVTL.W_S	0C94H
EP=095	CVTW.L_S	0C95H
EP=096	PLC.T_S	0C96H
EP=097	TZC_S	0C97H
EP=098	EQ.H_A	0C98H
EP=099	EQ.W_A	0C99H
EP=09a	EQ.H_JA	0C9AH
EP=09b	EQ.W_JA	0C9BH
EP=09c	EQ.B_S	0C9CH
EP=09d	EQ.H_S	0C9DH
EP=09e	EQ.W_S	0C9EH
EP=09f	EQ.L_S	0C9FH
EP=0a0	LEU.H_A	0CA0H
EP=0a1	LEU.W_A	0CA1H
EP=0a2	LEU.H_JA	0CA2H
EP=0a3	LEU.W_JA	0CA3H
EP=0a4	LEU.B_S	0CA4H
EP=0a5	LEU.H_S	0CA5H
EP=0a6	LEU.W_S	0CA6H
EP=0a7	LEU.L_S	0CA7H
EP=0a8	LTU.H_A	0CA8H
EP=0a9	LTU.W_A	0CA9H
EP=0aa	LTU.H_JA	0CAAH
EP=0ab	LTU.W_JA	0CABH
EP=0ac	LTU.B_S	0CACH
EP=0ad	LTU.H_S	0CADH
EP=0ae	LTU.W_S	0CAEH
EP=0af	LTU.L_S	0CAFH
EP=0b0	LE.H_A	0CB0H

EP=0b1	LE.W_A	0CB1H
EP=0b2	LE.H_JA	0CB2H
EP=0b3	LE.W_JA	0CB3H
EP=0b4	LE.B_S	0CB4H
EP=0b5	LE.H_S	0CB5H
EP=0b6	LE.W_S	0CB6H
EP=0b7	LE.L_S	0CB7H
EP=0b8	LT.H_A	0CB8H
EP=0b9	LT.W_A	0CB9H
EP=0ba	LT.H_JA	0CBAH
EP=0bb	LT.W_JA	0CBBH
EP=0bc	LT.B_S	0CBCH
EP=0bd	LT.H_S	0CBDH
EP=0be	LT.W_S	0CBEH
EP=0bf	LT.L_S	0CBFH
EP=0c0	ADD.W_SA	0CC0H
EP=0c1	SHF_A	0CC1H
EP=0c2	MOV_A	0CC2H
EP=0c3	MOV_SA	0CC3H
EP=0c4	MOV.W_S	0CC4H
EP=0c5	SHF_S	0CC5H
EP=0c6	MOV.L_S	0CC6H
EP=0c7	MOV_AS	0CC7H
EP=0c8	AND_A	0CC8H
EP=0c9	OR_A	0CC9H
EP=0ca	XOR_A	0CCAH
EP=0cb	NOT_A	0CCBH
EP=0cc	AND_S	0CCCH
EP=0cd	OR_S	0CCDH
EP=0ce	XOR_S	0CCEH
EP=0cf	NOT_S	0CCFH
EP=0d0	LE.S_S	0CD0H
EP=0d1	LE.D_S	0CD1H
EP=0d2	LT.S_S	0CD2H
EP=0d3	LT.D_S	0CD3H
EP=0d4	ADD.S_S	0CD4H
EP=0d5	ADD.D_S	0CD5H
EP=0d6	SUB.S_S	0CD6H
EP=0d7	SUB.D_S	0CD7H
EP=0d8	EQ.S_S	0CD8H
EP=0d9	EQ.D_S	0CD9H
EP=0da	NEG.H_A	0CDAH
EP=0db	NEG.W_A	0CDBH
EP=0dc	MUL.S_S	0CDCH
EP=0dd	MUL.D_S	0CDDH

EP=0de	DIV.S_S	0CDEH
EP=0df	DIV.D_S	0CDFH
EP=0e0	ADD.H_A	0CE0H
EP=0e1	ADD.W_A	0CE1H
EP=0e2	ADD.H_JA	0CE2H
EP=0e3	ADD.W_JA	0CE3H
EP=0e4	ADD.B_S	0CE4H
EP=0e5	ADD.H_S	0CE5H
EP=0e6	ADD.W_S	0CE6H
EP=0e7	ADD.L_S	0CE7H
EP=0e8	SUB.H_A	0CE8H
EP=0e9	SUB.W_A	0CE9H
EP=0ea	SUB.H_JA	0CEAH
EP=0eb	SUB.W_JA	0CEBH
EP=0ec	SUB.B_S	0CECH
EP=0ed	SUB.H_S	0CEDH
EP=0ee	SUB.W_S	0CEEH
EP=0ef	SUB.L_S	0CEFH
EP=0f0	MUL.H_A	0CF0H
EP=0f1	MUL.W_A	0CF1H
EP=0f2	MUL.H_JA	0CF2H
EP=0f3	MUL.W_JA	0CF3H
EP=0f4	MUL.B_S	0CF4H
EP=0f5	MUL.H_S	0CF5H
EP=0f6	MUL.W_S	0CF6H
EP=0f7	MUL.L_S	0CF7H
EP=0f8	DIV.H_A	0CF8H
EP=0f9	DIV.W_A	0CF9H
EP=0fa	DIV.H_JA	0CFAH
EP=0fb	DIV.W_JA	0CFBH
EP=0fc	DIV.B_S	0CFCH
EP=0fd	DIV.H_S	0CFDH
EP=0fe	DIV.W_S	0CFEH
EP=0ff	DIV.L_S	0CFFH
EP=100	MOV_VS	0D00H
EP=101	MOV_SV	0D01H
EP=102	MERG.T_V	0D02H
EP=103	MASK.T_V	0D03H
EP=104	MERG.F_SV	0D04H
EP=105	MASK.F_SV	0D05H
EP=106	MERG.T_SV	0D06H
EP=107	MASK.T_SV	0D07H
EP=108	MUL.S_V	0D08H
EP=109	MUL.D_V	0D09H
EP=10a	DIV.S_V	0D0AH

EP=10b	DIV.D_V	0D0BH
EP=10c	MUL.S_SV	0D0CH
EP=10d	MUL.D_SV	0D0DH
EP=10e	DIV.S_SV	0D0EH
EP=10f	DIV.D_SV	0D0FH
EP=110	AND_V	0D10H
EP=111	OR_V	0D11H
EP=112	XOR_V	0D12H
EP=113	SHF_V	0D13H
EP=114	AND_SV	0D14H
EP=115	OR_SV	0D15H
EP=116	XOR_SV	0D16H
EP=117	SHF_VSV	0D17H
EP=118	ADD.S_V	0D18H
EP=119	ADD.D_V	0D19H
EP=11a	SUB.S_V	0D1AH
EP=11b	SUB.D_V	0D1BH
EP=11c	ADD.S_SV	0D1CH
EP=11d	ADD.D_SV	0D1DH
EP=11e	SUB.S_SV	0D1EH
EP=11f	SUB.D_SV	0D1FH
EP=120	ADD.B_V	0D20H
EP=121	ADD.H_V	0D21H
EP=122	ADD.W_V	0D22H
EP=123	ADD.L_V	0D23H
EP=124	ADD.B_SV	0D24H
EP=125	ADD.H_SV	0D25H
EP=126	ADD.W_SV	0D26H
EP=127	ADD.L_SV	0D27H
EP=128	SUB.B_V	0D28H
EP=129	SUB.H_V	0D29H
EP=12a	SUB.W_V	0D2AH
EP=12b	SUB.L_V	0D2BH
EP=12c	SUB.B_SV	0D2CH
EP=12d	SUB.H_SV	0D2DH
EP=12e	SUB.W_SV	0D2EH
EP=12f	SUB.L_SV	0D2FH
EP=130	MUL.B_V	0D30H
EP=131	MUL.H_V	0D31H
EP=132	MUL.W_V	0D32H
EP=133	MUL.L_V	0D33H
EP=134	MUL.B_SV	0D34H
EP=135	MUL.H_SV	0D35H
EP=136	MUL.W_SV	0D36H
EP=137	MUL.L_SV	0D37H

EP=138	DIV.B_V	0D38H
EP=139	DIV.H_V	0D39H
EP=13a	DIV.W_V	0D3AH
EP=13b	DIV.L_V	0D3BH
EP=13c	DIV.B_SV	0D3CH
EP=13d	DIV.H_SV	0D3DH
EP=13e	DIV.W_SV	0D3EH
EP=13f	DIV.L_SV	0D3FH
EP=140	CVTD.S_V	0D40H
EP=141	CVTS.D_V	0D41H
EP=142	CVTL.D_V	0D42H
EP=143	CVTD.L_V	0D43H
EP=144	MOV_S_VM	0D44H
EP=145	MOV_VM_S	0D45H
EP=146	POWER.S_S	0D46H
EP=147	LOP_S	0D47H
EP=148	TZC_V	0D48H
EP=149	LOP_V	0D49H
EP=14a	POWER.D_S	0D4AH
EP=14b	NOT_V	0D4BH
EP=14c	SHF_SV	0D4CH
EP=14d	PLC.T_V	0D4DH
EP=14e	CPRS.F_V	0D4EH
EP=14f	CPRS.T_V	0D4FH
EP=150	EQ.S_V	0D50H
EP=151	EQ.D_V	0D51H
EP=152	NEG.S_V	0D52H
EP=153	NEG.D_V	0D53H
EP=154	EQ.S_SV	0D54H
EP=155	EQ.D_SV	0D55H
EP=156	NEG.S_S	0D56H
EP=157	NEG.D_S	0D57H
EP=158	LE.S_V	0D58H
EP=159	LE.D_V	0D59H
EP=15a	LT.S_V	0D5AH
EP=15b	LT.D_V	0D5BH
EP=15c	LE.S_SV	0D5CH
EP=15d	LE.D_SV	0D5DH
EP=15e	LT.S_SV	0D5EH
EP=15f	LT.D_SV	0D5FH
EP=160	EQ.B_V	0D60H
EP=161	EQ.H_V	0D61H
EP=162	EQ.W_V	0D62H
EP=163	EQ.L_V	0D63H
EP=164	EQ.B_SV	0D64H

EP=165	EQ.H_SV	0D65H
EP=166	EQ.W_SV	0D66H
EP=167	EQ.L_SV	0D67H
EP=168	LE.B_V	0D68H
EP=169	LE.H_V	0D69H
EP=16a	LE.W_V	0D6AH
EP=16b	LE.L_V	0D6BH
EP=16c	LE.B_SV	0D6CH
EP=16d	LE.H_SV	0D6DH
EP=16e	LE.W_SV	0D6EH
EP=16f	LE.L_SV	0D6FH
EP=170	LT.B_V	0D70H
EP=171	LT.H_V	0D71H
EP=172	LT.W_V	0D72H
EP=173	LT.L_V	0D73H
EP=174	LT.B_SV	0D74H
EP=175	LT.H_SV	0D75H
EP=176	LT.W_SV	0D76H
EP=177	LT.L_SV	0D77H
EP=178	NEG.B_V	0D78H
EP=179	NEG.H_V	0D79H
EP=17a	NEG.W_V	0D7AH
EP=17b	NEG.L_V	0D7BH
EP=17c	NEG.B_S	0D7CH
EP=17d	NEG.H_S	0D7DH
EP=17e	NEG.W_S	0D7EH
EP=17f	NEG.L_S	0D7FH
EP=180	LDSDR	0D80H
EP=181	LDKDR	0D81H
EP=182	LN.S_S	0D82H
EP=183	LN.D_S	0D83H
EP=184	PATU	0D84H
EP=185	PATE	0D85H
EP=186	PICH	0D86H
EP=187	PLCH	0D87H
EP=188	MOV_PSW_A	0D88H
EP=189	MOV_A_PSW	0D89H
EP=18a	MOV_PC_A	0D8AH
EP=18b	IDLE	0D8BH
EP=18e	STOP	0D8EH
EP=190	RTNQ	0D90H
EP=191	CFORK	0D91H
EP=192	RTN	0D92H
EP=193	WFORK	0D93H
EP=194	JOIN	0D94H

EP=195	RTNC	0D95H
EP=196	EXP.S_S	0D96H
EP=197	EXP.D_S	0D97H
EP=198	SIN.S_S	0D98H
EP=199	SIN.D_S	0D99H
EP=19a	ASIN.S_S	0D9AH
EP=19b	ASIN.D_S	0D9BH
EP=19c	COS.S_S	0D9CH
EP=19d	COS.D_S	0D9DH
EP=19e	ACOS.S_S	0D9EH
EP=19f	ACOS.D_S	0D9FH
EP=1a0	PSH.W_A	0DA0H
EP=1a1	LOG10.S_S	0DA1H
EP=1a2	POP.W_A	0DA2H
EP=1a3	LOG10.D_S	0DA3H
EP=1a4	PSH.W_S	0DA4H
EP=1a5	PSH.L_S	0DA5H
EP=1a6	POP.W_S	0DA6H
EP=1a7	POP.L_S	0DA7H
EP=1a8	ENI	0DA8H
EP=1a9	DSI	0DA9H
EP=1aa	BKPT	0DAAH
EP=1ab	MSYNC	0DABH
EP=1ac	MSKI	0DACH
EP=1ad	XMTI	0DADH
EP=1ae	MOV_S_VV	0DAEH
EP=1af	TSTVV	0DAFH
EP=1b0	MOV_VS_A	0DB0H
EP=1b1	MOV_A_VS	0DB1H
EP=1b2	MOV_VL_A	0DB2H
EP=1b3	MOV_A_VL	0DB3H
EP=1b4	MOV_VS_S	0DB4H
EP=1b5	MOV_S_VS	0DB5H
EP=1b6	MOV_VL_S	0DB6H
EP=1b7	MOV_S_VL	0DB7H
EP=1b8	DIAG	0DB8H
EP=1b9	PBKPT	0DB9H
EP=1ba	SQRT.S_S	0DBAH
EP=1bb	SQRT.D_S	0DBBH
EP=1bc	TAN.S_S	0DBCH
EP=1bd	TAN.D_S	0DBDH
EP=1be	ATAN.S_S	0DBEH
EP=1bf	ATAN.D_S	0DBFH
EP=1c0	SUM.B_SV	0DC0H
EP=1c1	SUM.H_SV	0DC1H

EP=1c2	SUM.W_SV	0DC2H
EP=1c3	SUM.L_SV	0DC3H
EP=1c4	ALL_SV	0DC4H
EP=1c5	ANY_SV	0DC5H
EP=1c6	PARITY_SV	0DC6H
EP=1c8	MAX.B_SV	0DC8H
EP=1c9	MAX.H_SV	0DC9H
EP=1ca	MAX.W_SV	0DCAH
EP=1cb	MAX.L_SV	0DCBH
EP=1cc	MIN.B_SV	0DCCH
EP=1cd	MIN.H_SV	0DCDH
EP=1ce	MIN.W_SV	0DCEH
EP=1cf	MIN.L_SV	0DCFH
EP=1d0	SUM.S_SV	0DD0H
EP=1d1	SUM.D_SV	0DD1H
EP=1d2	PROD.S_SV	0DD2H
EP=1d3	PROD.D_SV	0DD3H
EP=1d4	MAX.S_SV	0DD4H
EP=1d5	MAX.D_SV	0DD5H
EP=1d6	MIN.S_SV	0DD6H
EP=1d7	MIN.D_SV	0DD7H
EP=1d8	PROD.B_SV	0DD8H
EP=1d9	PROD.H_SV	0DD9H
EP=1da	PROD.W_SV	0DDAH
EP=1db	PROD.L_SV	0DDBH
EP=1dc	PLC.F_VM	0DDCH
EP=1dd	PLC.T_VM	0DDDH
EP=1de	ESCAPE_0	0DDEH
EP=1df	ESCAPE_1	0DDFH
EP=1e0	LDVI.B_V	0DE0H
EP=1e1	LDVI.H_V	0DE1H
EP=1e2	LDVI.W_V	0DE2H
EP=1e3	LDVI.L_V	0DE3H
EP=1e4	CVTW.S_V	0DE4H
EP=1e5	CVTS.W_V	0DE5H
EP=1e6	CVTW.L_V	0DE6H
EP=1e7	CVTL.W_V	0DE7H
EP=1e8	STVI.B_V	0DE8H
EP=1e9	STVI.H_V	0DE9H
EP=1ea	STVI.W_V	0DEAH
EP=1eb	STVI.L_V	0DEBH
EP=1ec	STVI.B_S	0DECH
EP=1ed	STVI.H_S	0DEDH
EP=1ee	STVI.W_S	0DEEH
EP=1ef	STVI.L_S	0DEFH

EP=1f0	NOP	0DF0H
EP=1f1	BR	0DF1H
EP=1f2	BRI.F	0DF2H
EP=1f3	BRI.T	0DF3H
EP=1f4	BRA.F	0DF4H
EP=1f5	BRA.T	0DF5H
EP=1f6	BRS.F	0DF6H
EP=1f7	BRS.T	0DF7H
EP=1f8	INTRPT	0DF8H
EP=1f9	TR_TRAP	0DF9H
EP=1fa	PROC_TRAP	0DFAH
EP=1fb	VV_TRAP	0DFBH
EP=1fc	TC_TRAP	0DFCH
EP=1fd	DL_TRAP	0DFDH
EP=1fe	AR_TRAP	0DFEH
EP=1ff	IDLE_TRAP	0DFFH
EP=202	TST	0E02H
EP=203	TST@	0E03H
EP=204	LCK	0E04H
EP=205	LCK@	0E05H
EP=206	ULK	0E06H
EP=207	ULK@	0E07H
EP=208	LDEA_S	0E08H
EP=209	LDEA_S@	0E09H
EP=20a	SPAWN	0E0AH
EP=20b	SPAWN@	0E0BH
EP=20c	LDCMR	0E0CH
EP=20d	LDCMR@	0E0DH
EP=20e	STCMR	0E0EH
EP=20f	STCMR@	0E0FH
EP=210	POPR_A	0E10H
EP=211	POPR_A@	0E11H
EP=212	PSHR_A	0E12H
EP=213	PSHR_A@	0E13H
EP=214	RCVR.W_A	0E14H
EP=215	RCVR.W_A@	0E15H
EP=216	MATM.W_A	0E16H
EP=217	MATM.W_A@	0E17H
EP=218	SNDR.W_A	0E18H
EP=219	SNDR.W_A@	0E19H
EP=21a	SNDR.L_S	0E1AH
EP=21b	SNDR.L_S@	0E1BH
EP=21c	RCVR.L_S	0E1CH
EP=21d	RCVR.L_S@	0E1DH
EP=21e	MATM.L_S	0E1EH

EP=21f	MATM.L_S@	0E1FH
EP=220	LD.D_IS	0E20H
EP=221	LD.U_IS	0E21H
EP=222	LD.L_IS	0E22H
EP=223	LD.W_IS	0E23H
EP=224	AND_IS	0E24H
EP=225	OR_IS	0E25H
EP=226	XOR_IS	0E26H
EP=227	SHF_IS	0E27H
EP=228	ADD.H_IS	0E28H
EP=229	ADD.W_IS	0E29H
EP=22a	SUB.H_IS	0E2AH
EP=22b	SUB.W_IS	0E2BH
EP=22c	MUL.H_IS	0E2CH
EP=22d	MUL.W_IS	0E2DH
EP=22e	DIV.H_IS	0E2EH
EP=22f	DIV.W_IS	0E2FH
EP=230	ADD.S_IS	0E30H
EP=231	SUB.S_IS	0E31H
EP=232	MUL.S_IS	0E32H
EP=233	DIV.S_IS	0E33H
EP=234	LE.S_IS	0E34H
EP=235	LT.S_IS	0E35H
EP=236	EQ.H_IS	0E36H
EP=237	EQ.W_IS	0E37H
EP=238	LEU.H_IS	0E38H
EP=239	LEU.W_IS	0E39H
EP=23a	LTU.H_IS	0E3AH
EP=23b	LTU.W_IS	0E3BH
EP=23c	LE.H_IS	0E3CH
EP=23d	LE.W_IS	0E3DH
EP=23e	LT.H_IS	0E3EH
EP=23f	LT.W_IS	0E3FH
EP=248	STE.B.X_S	0E48H
EP=249	STE.B.X_S@	0E49H
EP=24a	STE.H.X_S	0E4AH
EP=24b	STE.H.X_S@	0E4BH
EP=24c	STE.W.X_S	0E4CH
EP=24d	STE.W.X_S@	0E4DH
EP=24e	STE.L.X_S	0E4EH
EP=24f	STE.L.X_S@	0E4FH
EP=252	MAT.W_A	0E52H
EP=253	MAT.W_A@	0E53H
EP=254	GET.W_A	0E54H
EP=255	GET.W_A@	0E55H

EP=256	RCV.W_A	0E56H
EP=257	RCV.W_A@	0E57H
EP=25a	INC.W_A	0E5AH
EP=25b	INC.W_A@	0E5BH
EP=25c	PUT.W_A	0E5CH
EP=25d	PUT.W_A@	0E5DH
EP=25e	SND.W_A	0E5EH
EP=25f	SND.W_A@	0E5FH
EP=262	MAT.L_S	0E62H
EP=263	MAT.L_S@	0E63H
EP=264	GET.L_S	0E64H
EP=265	GET.L_S@	0E65H
EP=266	RCV.L_S	0E66H
EP=267	RCV.L_S@	0E67H
EP=26a	INC.L_S	0E6AH
EP=26b	INC.L_S@	0E6BH
EP=26c	PUT.L_S	0E6CH
EP=26d	PUT.L_S@	0E6DH
EP=26e	SND.L_S	0E6EH
EP=26f	SND.L_S@	0E6FH
EP=270	LD.B.X_V	0E70H
EP=271	LD.B.X_V@	0E71H
EP=272	LD.H.X_V	0E72H
EP=273	LD.H.X_V@	0E73H
EP=274	LD.W.X_V	0E74H
EP=275	LD.W.X_V@	0E75H
EP=276	LD.L.X_V	0E76H
EP=277	LD.L.X_V@	0E77H
EP=278	ST.B.X_V	0E78H
EP=279	ST.B.X_V@	0E79H
EP=27a	ST.H.X_V	0E7AH
EP=27b	ST.H.X_V@	0E7BH
EP=27c	ST.W.X_V	0E7CH
EP=27d	ST.W.X_V@	0E7DH
EP=27e	ST.L.X_V	0E7EH
EP=27f	ST.L.X_V@	0E7FH
EP=280	CVTW.B_V	0E80H
EP=281	CVTW.H_V	0E81H
EP=282	CVTB.W_V	0E82H
EP=283	CVTH.W_V	0E83H
EP=284	CVTW.B.X_V	0E84H
EP=285	CVTW.H.X_V	0E85H
EP=286	CVTB.W.X_V	0E86H
EP=287	CVTH.W.X_V	0E87H
EP=288	CVTS.L_V	0E88H

EP=289	CVTD.W_V	0E89H
EP=28a	CVTL.S_V	0E8AH
EP=28b	CVTW.D_V	0E8BH
EP=28c	CVTS.L.X_V	0E8CH
EP=28d	CVTD.W.X_V	0E8DH
EP=28e	CVTL.S.X_V	0E8EH
EP=28f	CVTW.D.X_V	0E8FH
EP=290	ENAL	0E90H
EP=291	SHF.W_S	0E91H
EP=292	ENAG	0E92H
EP=294	CVTW.D_S	0E94H
EP=295	CVTD.W_S	0E95H
EP=298	ATAN2.S_S	0E98H
EP=299	ATAN2.D_S	0E99H
EP=29c	FRINT.S_S	0E9CH
EP=29d	FRINT.D_S	0E9DH
EP=2e2	FRINT.S_V	0EE2H
EP=2e3	FRINT.D_V	0EE3H
EP=2e6	FRINT.S.X_V	0EE6H
EP=2e7	FRINT.D.X_V	0EE7H
EP=2f4	SQRT.S_V	0EF4H
EP=2f5	SQRT.D_V	0EF5H
EP=2fc	SQRT.S.X_V	0EFCH
EP=2fd	SQRT.D.X_V	0EFDH
EP=300	SUBR.S_SV	0F00H
EP=301	SUBR.D_SV	0F01H
EP=302	DIVR.S_SV	0F02H
EP=303	DIVR.D_SV	0F03H
EP=304	SUBR.S.X_SV	0F04H
EP=305	SUBR.D.X_SV	0F05H
EP=306	DIVR.S.X_SV	0F06H
EP=307	DIVR.D.X_SV	0F07H
EP=308	MUL.S.X_V	0F08H
EP=309	MUL.D.X_V	0F09H
EP=30a	DIV.S.X_V	0F0AH
EP=30b	DIV.D.X_V	0F0BH
EP=30c	MUL.S.X_SV	0F0CH
EP=30d	MUL.D.X_SV	0F0DH
EP=30e	DIV.S.X_SV	0F0EH
EP=30f	DIV.D.X_SV	0F0FH
EP=310	AND.X_V	0F10H
EP=311	OR.X_V	0F11H
EP=312	XOR.X_V	0F12H
EP=313	SHF.X_V	0F13H
EP=314	AND.X_SV	0F14H

EP=315	OR.X_SV	0F15H
EP=316	XOR.X_SV	0F16H
EP=317	SHF.X_VSV	0F17H
EP=318	ADD.S.X_V	0F18H
EP=319	ADD.D.X_V	0F19H
EP=31a	SUB.S.X_V	0F1AH
EP=31b	SUB.D.X_V	0F1BH
EP=31c	ADD.S.X_SV	0F1CH
EP=31d	ADD.D.X_SV	0F1DH
EP=31e	SUB.S.X_SV	0F1EH
EP=31f	SUB.D.X_SV	0F1FH
EP=320	ADD.B.X_V	0F20H
EP=321	ADD.H.X_V	0F21H
EP=322	ADD.W.X_V	0F22H
EP=323	ADD.L.X_V	0F23H
EP=324	ADD.B.X_SV	0F24H
EP=325	ADD.H.X_SV	0F25H
EP=326	ADD.W.X_SV	0F26H
EP=327	ADD.L.X_SV	0F27H
EP=328	SUB.B.X_V	0F28H
EP=329	SUB.H.X_V	0F29H
EP=32a	SUB.W.X_V	0F2AH
EP=32b	SUB.L.X_V	0F2BH
EP=32c	SUB.B.X_SV	0F2CH
EP=32d	SUB.H.X_SV	0F2DH
EP=32e	SUB.W.X_SV	0F2EH
EP=32f	SUB.L.X_SV	0F2FH
EP=330	MUL.B.X_V	0F30H
EP=331	MUL.H.X_V	0F31H
EP=332	MUL.W.X_V	0F32H
EP=333	MUL.L.X_V	0F33H
EP=334	MUL.B.X_SV	0F34H
EP=335	MUL.H.X_SV	0F35H
EP=336	MUL.W.X_SV	0F36H
EP=337	MUL.L.X_SV	0F37H
EP=338	DIV.B.X_V	0F38H
EP=339	DIV.H.X_V	0F39H
EP=33a	DIV.W.X_V	0F3AH
EP=33b	DIV.L.X_V	0F3BH
EP=33c	DIV.B.X_SV	0F3CH
EP=33d	DIV.H.X_SV	0F3DH
EP=33e	DIV.W.X_SV	0F3EH
EP=33f	DIV.L.X_SV	0F3FH
EP=340	CVTD.S.X_V	0F40H
EP=341	CVTS.D.X_V	0F41H

EP=342	CVTL.D.X_V	0F42H
EP=343	CVTD.L.X_V	0F43H
EP=348	TZC.X_V	0F48H
EP=349	LOP.X_V	0F49H
EP=34a	XPND.X_V	0F4AH
EP=34b	NOT.X_V	0F4BH
EP=34c	SHF.X_SV	0F4CH
EP=34d	PLC.T.X_V	0F4DH
EP=350	EQ.S.X_V	0F50H
EP=351	EQ.D.X_V	0F51H
EP=352	NEG.S.X_V	0F52H
EP=353	NEG.D.X_V	0F53H
EP=354	EQ.S.X_SV	0F54H
EP=355	EQ.D.X_SV	0F55H
EP=358	LE.S.X_V	0F58H
EP=359	LE.D.X_V	0F59H
EP=35a	LT.S.X_V	0F5AH
EP=35b	LT.D.X_V	0F5BH
EP=35c	LE.S.X_SV	0F5CH
EP=35d	LE.D.X_SV	0F5DH
EP=35e	LT.S.X_SV	0F5EH
EP=35f	LT.D.X_SV	0F5FH
EP=360	EQ.B.X_V	0F60H
EP=361	EQ.H.X_V	0F61H
EP=362	EQ.W.X_V	0F62H
EP=363	EQ.L.X_V	0F63H
EP=364	EQ.B.X_SV	0F64H
EP=365	EQ.H.X_SV	0F65H
EP=366	EQ.W.X_SV	0F66H
EP=367	EQ.L.X_SV	0F67H
EP=368	LE.B.X_V	0F68H
EP=369	LE.H.X_V	0F69H
EP=36a	LE.W.X_V	0F6AH
EP=36b	LE.L.X_V	0F6BH
EP=36c	LE.B.X_SV	0F6CH
EP=36d	LE.H.X_SV	0F6DH
EP=36e	LE.W.X_SV	0F6EH
EP=36f	LE.L.X_SV	0F6FH
EP=370	LT.B.X_V	0F70H
EP=371	LT.H.X_V	0F71H
EP=372	LT.W.X_V	0F72H
EP=373	LT.L.X_V	0F73H
EP=374	LT.B.X_SV	0F74H
EP=375	LT.H.X_SV	0F75H
EP=376	LT.W.X_SV	0F76H

EP=377	LT.L.X_SV	0F77H
EP=378	NEG.B.X_V	0F78H
EP=379	NEG.H.X_V	0F79H
EP=37a	NEG.W.X_V	0F7AH
EP=37b	NEG.L.X_V	0F7BH
EP=380	MOV_S_CIR	0F80H
EP=381	MOV_CIR_S	0F81H
EP=382	MOV_TOC_S	0F82H
EP=383	MOV_CPUID_S	0F83H
EP=384	MOV_S_TTR	0F84H
EP=385	MOV_TTR_S	0F85H
EP=386	CTRSL	0F86H
EP=387	CTRSG	0F87H
EP=388	MOV_S_VMU	0F88H
EP=389	MOV_VMU_S	0F89H
EP=38a	MOV_S_VML	0F8AH
EP=38b	MOV_VML_S	0F8BH
EP=38c	MOV_S_ICR	0F8CH
EP=38d	MOV_ICR_S	0F8DH
EP=38e	MOV_S_TCPU	0F8EH
EP=38f	MOV_TCPU_S	0F8FH
EP=396	MOV_S_TID	0F96H
EP=397	MOV_TID_S	0F97H
EP=3c0	SUM.B.X_SV	0FC0H
EP=3c1	SUM.H.X_SV	0FC1H
EP=3c2	SUM.W.X_SV	0FC2H
EP=3c3	SUM.L.X_SV	0FC3H
EP=3c4	ALL.X_SV	0FC4H
EP=3c5	ANY.X_SV	0FC5H
EP=3c6	PARITY.X_SV	0FC6H
EP=3c8	MAX.B.X_SV	0FC8H
EP=3c9	MAX.H.X_SV	0FC9H
EP=3ca	MAX.W.X_SV	0FCAH
EP=3cb	MAX.L.X_SV	0FCBH
EP=3cc	MIN.B.X_SV	0FCCH
EP=3cd	MIN.H.X_SV	0FCDH
EP=3ce	MIN.W.X_SV	0FCEH
EP=3cf	MIN.L.X_SV	0FCFH
EP=3d0	SUM.S.X_SV	0FD0H
EP=3d1	SUM.D.X_SV	0FD1H
EP=3d2	PROD.S.X_SV	0FD2H
EP=3d3	PROD.D.X_SV	0FD3H
EP=3d4	MAX.S.X_SV	0FD4H
EP=3d5	MAX.D.X_SV	0FD5H
EP=3d6	MIN.S.X_SV	0FD6H

EP=3d7	MIN.D.X_SV	0FD7H
EP=3d8	PROD.B.X_SV	0FD8H
EP=3d9	PROD.H.X_SV	0FD9H
EP=3da	PROD.W.X_SV	0FDAH
EP=3db	PROD.L.X_SV	0FDBH
EP=3e0	LDVI.B.X_V	0FE0H
EP=3e1	LDVI.H.X_V	0FE1H
EP=3e2	LDVI.W.X_V	0FE2H
EP=3e3	LDVI.L.X_V	0FE3H
EP=3e4	CVTW.S.X_V	0FE4H
EP=3e5	CVTS.W.X_V	0FE5H
EP=3e6	CVTW.L.X_V	0FE6H
EP=3e7	CVTL.W.X_V	0FE7H
EP=3e8	STVI.B.X_V	0FE8H
EP=3e9	STVI.H.X_V	0FE9H
EP=3ea	STVI.W.X_V	0FEAH
EP=3eb	STVI.L.X_V	0FEBH
EP=3ec	STVI.B.X_S	0FECH
EP=3ed	STVI.H.X_S	0FEDH
EP=3ee	STVI.W.X_S	0FEEH
EP=3ef	STVI.L.X_S	0FEFH

C.3 ASP Entry Points — by Opcode

Opcode	UPC	Entry Point
ACOS.D_S	0D9FH	EP=19f
ACOS.S_S	0D9EH	EP=19e
ADD.B_S	0CE4H	EP=0e4
ADD.B_SV	0D24H	EP=124
ADD.B_V	0D20H	EP=120
ADD.B.X_SV	0F24H	EP=324
ADD.B.X_V	0F20H	EP=320
ADD.D_S	0CD5H	EP=0d5
ADD.D_SV	0D1DH	EP=11d
ADD.D_V	0D19H	EP=119
ADD.D.X_SV	0F1DH	EP=31d
ADD.D.X_V	0F19H	EP=319
ADD.H_A	0CE0H	EP=0e0
ADD.H_IA	0C28H	EP=028
ADD.H_IS	0E28H	EP=228
ADD.H_JA	0CE2H	EP=0e2
ADD.H_S	0CE5H	EP=0e5
ADD.H_SV	0D25H	EP=125
ADD.H_V	0D21H	EP=121
ADD.H.X_SV	0F25H	EP=325
ADD.H.X_V	0F21H	EP=321
ADD.L_S	0CE7H	EP=0e7
ADD.L_SV	0D27H	EP=127
ADD.L_V	0D23H	EP=123
ADD.L.X_SV	0F27H	EP=327
ADD.L.X_V	0F23H	EP=323
ADD.S_IS	0E30H	EP=230
ADD.S_S	0CD4H	EP=0d4
ADD.S_SV	0D1CH	EP=11c
ADD.S_V	0D18H	EP=118
ADD.S.X_SV	0F1CH	EP=31c
ADD.S.X_V	0F18H	EP=318
ADD.W_A	0CE1H	EP=0e1
ADD.W_IA	0C29H	EP=029
ADD.W_IS	0E29H	EP=229
ADD.W_JA	0CE3H	EP=0e3
ADD.W_S	0CE6H	EP=0e6
ADD.W_SA	0CC0H	EP=0c0
ADD.W_SV	0D26H	EP=126
ADD.W_V	0D22H	EP=122

ADD.W.X_SV	0F26H	EP=326
ADD.W.X_V	0F22H	EP=322
ALL_SV	0DC4H	EP=1c4
ALL.X_SV	0FC4H	EP=3c4
AND_A	0CC8H	EP=0c8
AND_IA	0C24H	EP=024
AND_IS	0E24H	EP=224
AND_S	0CCCH	EP=0cc
AND_SV	0D14H	EP=114
AND_V	0D10H	EP=110
AND.X_SV	0F14H	EP=314
AND.X_V	0F10H	EP=310
ANY_SV	0DC5H	EP=1c5
ANY.X_SV	0FC5H	EP=3c5
AR_TRAP	0DFEH	EP=1fe
ASIN.D_S	0D9BH	EP=19b
ASIN.S_S	0D9AH	EP=19a
ATAN2.D_S	0E99H	EP=299
ATAN2.S_S	0E98H	EP=298
ATAN.D_S	0DBFH	EP=1bf
ATAN.S_S	0DBEH	EP=1be
BKPT	0DAAH	EP=1aa
BR	0DF1H	EP=1f1
BRA.F	0DF4H	EP=1f4
BRA.T	0DF5H	EP=1f5
BRI.F	0DF2H	EP=1f2
BRI.T	0DF3H	EP=1f3
BRS.F	0DF6H	EP=1f6
BRS.T	0DF7H	EP=1f7
CALL	0C40H	EP=040
CALL@	0C41H	EP=041
CALLQ	0C44H	EP=044
CALLQ@	0C45H	EP=045
CALLS	0C42H	EP=042
CALLS@	0C43H	EP=043
CFORK	0D91H	EP=191
COS.D_S	0D9DH	EP=19d
COS.S_S	0D9CH	EP=19c
CPRS.F_V	0D4EH	EP=14e
CPRS.T_V	0D4FH	EP=14f
CTRSG	0F87H	EP=387
CTRSL	0F86H	EP=386
CVTB.W_A	0C82H	EP=082
CVTB.W_S	0C86H	EP=086
CVTB.W_V	0E82H	EP=282

CVTB.W.X_V	0E86H	EP=286
CVTD.L_S	0C8DH	EP=08d
CVTD.L_V	0D43H	EP=143
CVTD.L.X_V	0F43H	EP=343
CVTD.S_S	0C8AH	EP=08a
CVTD.S_V	0D40H	EP=140
CVTD.S.X_V	0F40H	EP=340
CVTD.W_S	0E95H	EP=295
CVTD.W_V	0E89H	EP=289
CVTD.W.X_V	0E8DH	EP=28d
CVTH.W_A	0C83H	EP=083
CVTH.W_S	0C87H	EP=087
CVTH.W_V	0E83H	EP=283
CVTH.W.X_V	0E87H	EP=287
CVTL.D_S	0C8FH	EP=08f
CVTL.D_V	0D42H	EP=142
CVTL.D.X_V	0F42H	EP=342
CVTL.S_S	0C8EH	EP=08e
CVTL.S_V	0E8AH	EP=28a
CVTL.S.X_V	0E8EH	EP=28e
CVTL.W_S	0C94H	EP=094
CVTL.W_V	0DE7H	EP=1e7
CVTL.W.X_V	0FE7H	EP=3e7
CVTS.D_S	0C8BH	EP=08b
CVTS.D_V	0D41H	EP=141
CVTS.D.X_V	0F41H	EP=341
CVTS.L_S	0C8CH	EP=08c
CVTS.L_V	0E88H	EP=288
CVTS.L.X_V	0E8CH	EP=28c
CVTS.W_S	0C89H	EP=089
CVTS.W_V	0DE5H	EP=1e5
CVTS.W.X_V	0FE5H	EP=3e5
CVTW.B_A	0C80H	EP=080
CVTW.B_S	0C84H	EP=084
CVTW.B_V	0E80H	EP=280
CVTW.B.X_V	0E84H	EP=284
CVTW.D_S	0E94H	EP=294
CVTW.D_V	0E8BH	EP=28b
CVTW.D.X_V	0E8FH	EP=28f
CVTW.H_A	0C81H	EP=081
CVTW.H_S	0C85H	EP=085
CVTW.H_V	0E81H	EP=281
CVTW.H.X_V	0E85H	EP=285
CVTW.L_S	0C95H	EP=095
CVTW.L_V	0DE6H	EP=1e6

CVTW.L.X_V	0FE6H	EP=3e6
CVTW.S_S	0C88H	EP=088
CVTW.S_V	0DE4H	EP=1e4
CVTW.S.X_V	0FE4H	EP=3e4
DIAG	0DB8H	EP=1b8
DIV.B_S	0CFCH	EP=0fc
DIV.B_SV	0D3CH	EP=13c
DIV.B_V	0D38H	EP=138
DIV.B.X_SV	0F3CH	EP=33c
DIV.B.X_V	0F38H	EP=338
DIV.D_S	0CDFH	EP=0df
DIV.D_SV	0D0FH	EP=10f
DIV.D_V	0D0BH	EP=10b
DIV.D.X_SV	0F0FH	EP=30f
DIV.D.X_V	0F0BH	EP=30b
DIV.H_A	0CF8H	EP=0f8
DIV.H_IA	0C2EH	EP=02e
DIV.H_IS	0E2EH	EP=22e
DIV.H_JA	0CFAH	EP=0fa
DIV.H_S	0CFDH	EP=0fd
DIV.H_SV	0D3DH	EP=13d
DIV.H_V	0D39H	EP=139
DIV.H.X_SV	0F3DH	EP=33d
DIV.H.X_V	0F39H	EP=339
DIV.L_S	0CFFH	EP=0ff
DIV.L_SV	0D3FH	EP=13f
DIV.L_V	0D3BH	EP=13b
DIV.L.X_SV	0F3FH	EP=33f
DIV.L.X_V	0F3BH	EP=33b
DIVR.D_SV	0F03H	EP=303
DIVR.D.X_SV	0F07H	EP=307
DIVR.S_SV	0F02H	EP=302
DIVR.S.X_SV	0F06H	EP=306
DIV.S_IS	0E33H	EP=233
DIV.S_S	0CDEH	EP=0de
DIV.S_SV	0D0EH	EP=10e
DIV.S_V	0D0AH	EP=10a
DIV.S.X_SV	0F0EH	EP=30e
DIV.S.X_V	0F0AH	EP=30a
DIV.W_A	0CF9H	EP=0f9
DIV.W_IA	0C2FH	EP=02f
DIV.W_IS	0E2FH	EP=22f
DIV.W_JA	0CFBH	EP=0fb
DIV.W_S	0CFEH	EP=0fe
DIV.W_SV	0D3EH	EP=13e

DIV.W_V	0D3AH	EP=13a
DIV.W.X_SV	0F3EH	EP=33e
DIV.W.X_V	0F3AH	EP=33a
DL_TRAP	0DFDH	EP=1fd
DSI	0DA9H	EP=1a9
ENAG	0E92H	EP=292
ENAL	0E90H	EP=290
ENI	0DA8H	EP=1a8
EQ.B_S	0C9CH	EP=09c
EQ.B_SV	0D64H	EP=164
EQ.B_V	0D60H	EP=160
EQ.B.X_SV	0F64H	EP=364
EQ.B.X_V	0F60H	EP=360
EQ.D_S	0CD9H	EP=0d9
EQ.D_SV	0D55H	EP=155
EQ.D_V	0D51H	EP=151
EQ.D.X_SV	0F55H	EP=355
EQ.D.X_V	0F51H	EP=351
EQ.H_A	0C98H	EP=098
EQ.H_IA	0C36H	EP=036
EQ.H_IS	0E36H	EP=236
EQ.H_JA	0C9AH	EP=09a
EQ.H_S	0C9DH	EP=09d
EQ.H_SV	0D65H	EP=165
EQ.H_V	0D61H	EP=161
EQ.H.X_SV	0F65H	EP=365
EQ.H.X_V	0F61H	EP=361
EQ.L_S	0C9FH	EP=09f
EQ.L_SV	0D67H	EP=167
EQ.L_V	0D63H	EP=163
EQ.L.X_SV	0F67H	EP=367
EQ.L.X_V	0F63H	EP=363
EQ.S_S	0CD8H	EP=0d8
EQ.S_SV	0D54H	EP=154
EQ.S_V	0D50H	EP=150
EQ.S.X_SV	0F54H	EP=354
EQ.S.X_V	0F50H	EP=350
EQ.W_A	0C99H	EP=099
EQ.W_IA	0C37H	EP=037
EQ.W_IS	0E37H	EP=237
EQ.W_JA	0C9BH	EP=09b
EQ.W_S	0C9EH	EP=09e
EQ.W_SV	0D66H	EP=166
EQ.W_V	0D62H	EP=162
EQ.W.X_SV	0F66H	EP=366

EQ.W.X_V	0F62H	EP=362
ESCAPE_0	0DDEH	EP=1de
ESCAPE_1	0DDFH	EP=1df
EXIT	0C00H	EP=000
EXIT@	0C01H	EP=001
EXP.D_S	0D97H	EP=197
EXP.S_S	0D96H	EP=196
FRINT.D_S	0E9DH	EP=29d
FRINT.D_V	0EE3H	EP=2e3
FRINT.D.X_V	0EE7H	EP=2e7
FRINT.S_S	0E9CH	EP=29c
FRINT.S_V	0EE2H	EP=2e2
FRINT.S.X_V	0EE6H	EP=2e6
GET.L_S	0E64H	EP=264
GET.L_S@	0E65H	EP=265
GET.W_A	0E54H	EP=254
GET.W_A@	0E55H	EP=255
HALT	0C20H	EP=020
IDLE	0D8BH	EP=18b
IDLE_TRAP	0DFFH	EP=1ff
INC.L_S	0E6AH	EP=26a
INC.L_S@	0E6BH	EP=26b
INCR.L_S	0C5EH	EP=05e
INCR.L_S@	0C5FH	EP=05f
INCR.W_A	0C56H	EP=056
INCR.W_A@	0C57H	EP=057
INC.W_A	0E5AH	EP=25a
INC.W_A@	0E5BH	EP=25b
INTRPT	0DF8H	EP=1f8
JMP	0C02H	EP=002
JMP@	0C03H	EP=003
JMPA.F	0C08H	EP=008
JMPA.F@	0C09H	EP=009
JMPA.T	0C0AH	EP=00a
JMPA.T@	0C0BH	EP=00b
JMPI.F	0C04H	EP=004
JMPI.F@	0C05H	EP=005
JMPI.T	0C06H	EP=006
JMPI.T@	0C07H	EP=007
JMPS.F	0C0CH	EP=00c
JMPS.F@	0C0DH	EP=00d
JMPS.T	0C0EH	EP=00e
JMPS.T@	0C0FH	EP=00f
JOIN	0D94H	EP=194
LCK	0E04H	EP=204

LCK@	0E05H	EP=205
LD.B_A	0C50H	EP=050
LD.B_A@	0C51H	EP=051
LD.B_S	0C60H	EP=060
LD.B_S@	0C61H	EP=061
LD.B_V	0C70H	EP=070
LD.B_V@	0C71H	EP=071
LD.B.X_V	0E70H	EP=270
LD.B.X_V@	0E71H	EP=271
LDCMR	0E0CH	EP=20c
LDCMR@	0E0DH	EP=20d
LD.D_IS	0E20H	EP=220
LDEA_A	0C12H	EP=012
LDEA_A@	0C13H	EP=013
LDEA_S	0E08H	EP=208
LDEA_S@	0E09H	EP=209
LD.H_A	0C52H	EP=052
LD.H_A@	0C53H	EP=053
LD.H_IA	0C22H	EP=022
LD.H_JA	0C92H	EP=092
LD.H_S	0C62H	EP=062
LD.H_S@	0C63H	EP=063
LD.H_V	0C72H	EP=072
LD.H_V@	0C73H	EP=073
LD.H.X_V	0E72H	EP=272
LD.H.X_V@	0E73H	EP=273
LD_IVL	0C30H	EP=030
LD_IVS	0C31H	EP=031
LDKDR	0D81H	EP=181
LD.L_IS	0E22H	EP=222
LD.L_S	0C66H	EP=066
LD.L_S@	0C67H	EP=067
LD.L_V	0C76H	EP=076
LD.L_V@	0C77H	EP=077
LD.L_VLS	0C14H	EP=014
LD.L_VLS@	0C15H	EP=015
LD.L.X_V	0E76H	EP=276
LD.L.X_V@	0E77H	EP=277
LDPA	0C90H	EP=090
LDSDR	0D80H	EP=180
LD.U_IS	0E21H	EP=221
LDVI.B_V	0DE0H	EP=1e0
LDVI.B.X_V	0FE0H	EP=3e0
LDVI.H_V	0DE1H	EP=1e1
LDVI.H.X_V	0FE1H	EP=3e1

LDVI.L_V	0DE3H	EP=1e3
LDVI.L.X_V	0FE3H	EP=3e3
LDVI.W_V	0DE2H	EP=1e2
LDVI.W.X_V	0FE2H	EP=3e2
LD.W_A	0C54H	EP=054
LD.W_A@	0C55H	EP=055
LD.W_IA	0C23H	EP=023
LD.W_IS	0E23H	EP=223
LD.W_JA	0C93H	EP=093
LD.W_S	0C64H	EP=064
LD.W_S@	0C65H	EP=065
LD.W_V	0C74H	EP=074
LD.W_V@	0C75H	EP=075
LD.W.X_V	0E74H	EP=274
LD.W.X_V@	0E75H	EP=275
LD.X_VM	0C16H	EP=016
LD.X_VM@	0C17H	EP=017
LE.B_S	0CB4H	EP=0b4
LE.B_SV	0D6CH	EP=16c
LE.B_V	0D68H	EP=168
LE.B.X_SV	0F6CH	EP=36c
LE.B.X_V	0F68H	EP=368
LE.D_S	0CD1H	EP=0d1
LE.D_SV	0D5DH	EP=15d
LE.D_V	0D59H	EP=159
LE.D.X_SV	0F5DH	EP=35d
LE.D.X_V	0F59H	EP=359
LE.H_A	0CB0H	EP=0b0
LE.H_IA	0C3CH	EP=03c
LE.H_IS	0E3CH	EP=23c
LE.H_JA	0CB2H	EP=0b2
LE.H_S	0CB5H	EP=0b5
LE.H_SV	0D6DH	EP=16d
LE.H_V	0D69H	EP=169
LE.H.X_SV	0F6DH	EP=36d
LE.H.X_V	0F69H	EP=369
LE.L_S	0CB7H	EP=0b7
LE.L_SV	0D6FH	EP=16f
LE.L_V	0D6BH	EP=16b
LE.L.X_SV	0F6FH	EP=36f
LE.L.X_V	0F6BH	EP=36b
LE.S_IS	0E34H	EP=234
LE.S_S	0CD0H	EP=0d0
LE.S_SV	0D5CH	EP=15c
LE.S_V	0D58H	EP=158

LE.S.X_SV	0F5CH	EP=35c
LE.S.X_V	0F58H	EP=358
LEU.B_S	0CA4H	EP=0a4
LEU.H_A	0CA0H	EP=0a0
LEU.H_IA	0C38H	EP=038
LEU.H_IS	0E38H	EP=238
LEU.H_JA	0CA2H	EP=0a2
LEU.H_S	0CA5H	EP=0a5
LEU.L_S	0CA7H	EP=0a7
LEU.W_A	0CA1H	EP=0a1
LEU.W_IA	0C39H	EP=039
LEU.W_IS	0E39H	EP=239
LEU.W_JA	0CA3H	EP=0a3
LEU.W_S	0CA6H	EP=0a6
LE.W_A	0CB1H	EP=0b1
LE.W_IA	0C3DH	EP=03d
LE.W_IS	0E3DH	EP=23d
LE.W_JA	0CB3H	EP=0b3
LE.W_S	0CB6H	EP=0b6
LE.W_SV	0D6EH	EP=16e
LE.W_V	0D6AH	EP=16a
LE.W.X_SV	0F6EH	EP=36e
LE.W.X_V	0F6AH	EP=36a
LN.D_S	0D83H	EP=183
LN.S_S	0D82H	EP=182
LOG10.D_S	0DA3H	EP=1a3
LOG10.S_S	0DA1H	EP=1a1
LOP_S	0D47H	EP=147
LOP_V	0D49H	EP=149
LOP.X_V	0F49H	EP=349
LT.B_S	0CBCH	EP=0bc
LT.B_SV	0D74H	EP=174
LT.B_V	0D70H	EP=170
LT.B.X_SV	0F74H	EP=374
LT.B.X_V	0F70H	EP=370
LT.D_S	0CD3H	EP=0d3
LT.D_SV	0D5FH	EP=15f
LT.D_V	0D5BH	EP=15b
LT.D.X_SV	0F5FH	EP=35f
LT.D.X_V	0F5BH	EP=35b
LT.H_A	0CB8H	EP=0b8
LT.H_IA	0C3EH	EP=03e
LT.H_IS	0E3EH	EP=23e
LT.H_JA	0CBAH	EP=0ba
LT.H_S	0CBDH	EP=0bd

LT.H_SV	0D75H	EP=175
LT.H_V	0D71H	EP=171
LT.H.X_SV	0F75H	EP=375
LT.H.X_V	0F71H	EP=371
LT.L_S	0CBFH	EP=0bf
LT.L_SV	0D77H	EP=177
LT.L_V	0D73H	EP=173
LT.L.X_SV	0F77H	EP=377
LT.L.X_V	0F73H	EP=373
LT.S_IS	0E35H	EP=235
LT.S_S	0CD2H	EP=0d2
LT.S_SV	0D5EH	EP=15e
LT.S_V	0D5AH	EP=15a
LT.S.X_SV	0F5EH	EP=35e
LT.S.X_V	0F5AH	EP=35a
LTU.B_S	0CACH	EP=0ac
LTU.H_A	0CA8H	EP=0a8
LTU.H_IA	0C3AH	EP=03a
LTU.H_IS	0E3AH	EP=23a
LTU.H_JA	0CAAH	EP=0aa
LTU.H_S	0CADH	EP=0ad
LTU.L_S	0CAFH	EP=0af
LTU.W_A	0CA9H	EP=0a9
LTU.W_IA	0C3BH	EP=03b
LTU.W_IS	0E3BH	EP=23b
LTU.W_JA	0CABH	EP=0ab
LTU.W_S	0CAEH	EP=0ae
LT.W_A	0CB9H	EP=0b9
LT.W_IA	0C3FH	EP=03f
LT.W_IS	0E3FH	EP=23f
LT.W_JA	0CBBH	EP=0bb
LT.W_S	0CBEH	EP=0be
LT.W_SV	0D76H	EP=176
LT.W_V	0D72H	EP=172
LT.W.X_SV	0F76H	EP=376
LT.W.X_V	0F72H	EP=372
MASK.F_SV	0D05H	EP=105
MASK.T_SV	0D07H	EP=107
MASK.T_V	0D03H	EP=103
MAT.L_S	0E62H	EP=262
MAT.L_S@	0E63H	EP=263
MATM.L_S	0E1EH	EP=21e
MATM.L_S@	0E1FH	EP=21f
MATM.W_A	0E16H	EP=216
MATM.W_A@	0E17H	EP=217

MAT.W_A	0E52H	EP=252
MAT.W_A@	0E53H	EP=253
MAX.B_SV	0DC8H	EP=1c8
MAX.B.X_SV	0FC8H	EP=3c8
MAX.D_SV	0DD5H	EP=1d5
MAX.D.X_SV	0FD5H	EP=3d5
MAX.H_SV	0DC9H	EP=1c9
MAX.H.X_SV	0FC9H	EP=3c9
MAX.L_SV	0DCBH	EP=1cb
MAX.L.X_SV	0FCBH	EP=3cb
MAX.S_SV	0DD4H	EP=1d4
MAX.S.X_SV	0FD4H	EP=3d4
MAX.W_SV	0DCAH	EP=1ca
MAX.W.X_SV	0FCAH	EP=3ca
MERG.F_SV	0D04H	EP=104
MERG.T_SV	0D06H	EP=106
MERG.T_V	0D02H	EP=102
MIN.B_SV	0DCCH	EP=1cc
MIN.B.X_SV	0FCCH	EP=3cc
MIN.D_SV	0DD7H	EP=1d7
MIN.D.X_SV	0FD7H	EP=3d7
MIN.H_SV	0DCDH	EP=1cd
MIN.H.X_SV	0FCDH	EP=3cd
MIN.L_SV	0DCFH	EP=1cf
MIN.L.X_SV	0FCFH	EP=3cf
MIN.S_SV	0DD6H	EP=1d6
MIN.S.X_SV	0FD6H	EP=3d6
MIN.W_SV	0DCEH	EP=1ce
MIN.W.X_SV	0FCEH	EP=3ce
MOV_A	0CC2H	EP=0c2
MOV_A_PSW	0D89H	EP=189
MOV_AS	0CC7H	EP=0c7
MOV_A_VL	0DB3H	EP=1b3
MOV_A_VS	0DB1H	EP=1b1
MOV_CIR_S	0F81H	EP=381
MOV_CPUID_S	0F83H	EP=383
MOV_ICR_S	0F8DH	EP=38d
MOV.L_S	0CC6H	EP=0c6
MOV_PC_A	0D8AH	EP=18a
MOV_PSW_A	0D88H	EP=188
MOV_SA	0CC3H	EP=0c3
MOV_S_CIR	0F80H	EP=380
MOV_S_ICR	0F8CH	EP=38c
MOV_S_TCPU	0F8EH	EP=38e
MOV_S_TID	0F96H	EP=396

MOV_S_TTR	0F84H	EP=384
MOV_SV	0D01H	EP=101
MOV_S_VL	0DB7H	EP=1b7
MOV_S_VM	0D44H	EP=144
MOV_S_VML	0F8AH	EP=38a
MOV_S_VMU	0F88H	EP=388
MOV_S_VS	0DB5H	EP=1b5
MOV_S_VV	0DAEH	EP=1ae
MOV_TCPU_S	0F8FH	EP=38f
MOV_TID_S	0F97H	EP=397
MOV_TOC_S	0F82H	EP=382
MOV_TTR_S	0F85H	EP=385
MOV_VL_A	0DB2H	EP=1b2
MOV_VL_S	0DB6H	EP=1b6
MOV_VML_S	0F8BH	EP=38b
MOV_VM_S	0D45H	EP=145
MOV_VMU_S	0F89H	EP=389
MOV_VS	0D00H	EP=100
MOV_VS_A	0DB0H	EP=1b0
MOV_VS_S	0DB4H	EP=1b4
MOV.W_S	0CC4H	EP=0c4
MSKI	0DACH	EP=1ac
MSYNC	0DABH	EP=1ab
MUL.B_S	0CF4H	EP=0f4
MUL.B_SV	0D34H	EP=134
MUL.B_V	0D30H	EP=130
MUL.B.X_SV	0F34H	EP=334
MUL.B.X_V	0F30H	EP=330
MUL.D_S	0CDDH	EP=0dd
MUL.D_SV	0D0DH	EP=10d
MUL.D_V	0D09H	EP=109
MUL.D.X_SV	0F0DH	EP=30d
MUL.D.X_V	0F09H	EP=309
MUL.H_A	0CF0H	EP=0f0
MUL.H_IA	0C2CH	EP=02c
MUL.H_IS	0E2CH	EP=22c
MUL.H_JA	0CF2H	EP=0f2
MUL.H_S	0CF5H	EP=0f5
MUL.H_SV	0D35H	EP=135
MUL.H_V	0D31H	EP=131
MUL.H.X_SV	0F35H	EP=335
MUL.H.X_V	0F31H	EP=331
MUL.L_S	0CF7H	EP=0f7
MUL.L_SV	0D37H	EP=137
MUL.L_V	0D33H	EP=133

MUL.L.X_SV	0F37H	EP=337
MUL.L.X_V	0F33H	EP=333
MUL.S_IS	0E32H	EP=232
MUL.S_S	0CDCH	EP=0dc
MUL.S_SV	0D0CH	EP=10c
MUL.S_V	0D08H	EP=108
MUL.S.X_SV	0F0CH	EP=30c
MUL.S.X_V	0F08H	EP=308
MUL.W_A	0CF1H	EP=0f1
MUL.W_IA	0C2DH	EP=02d
MUL.W_IS	0E2DH	EP=22d
MUL.W_JA	0CF3H	EP=0f3
MUL.W_S	0CF6H	EP=0f6
MUL.W_SV	0D36H	EP=136
MUL.W_V	0D32H	EP=132
MUL.W.X_SV	0F36H	EP=336
MUL.W.X_V	0F32H	EP=332
NEG.B_S	0D7CH	EP=17c
NEG.B_V	0D78H	EP=178
NEG.B.X_V	0F78H	EP=378
NEG.D_S	0D57H	EP=157
NEG.D_V	0D53H	EP=153
NEG.D.X_V	0F53H	EP=353
NEG.H_A	0CDAH	EP=0da
NEG.H_S	0D7DH	EP=17d
NEG.H_V	0D79H	EP=179
NEG.H.X_V	0F79H	EP=379
NEG.L_S	0D7FH	EP=17f
NEG.L_V	0D7BH	EP=17b
NEG.L.X_V	0F7BH	EP=37b
NEG.S_S	0D56H	EP=156
NEG.S_V	0D52H	EP=152
NEG.S.X_V	0F52H	EP=352
NEG.W_A	0CDBH	EP=0db
NEG.W_S	0D7EH	EP=17e
NEG.W_V	0D7AH	EP=17a
NEG.W.X_V	0F7AH	EP=37a
NOP	0DF0H	EP=1f0
NOT_A	0CCBH	EP=0cb
NOT_S	0CCFH	EP=0cf
NOT_V	0D4BH	EP=14b
NOT.X_V	0F4BH	EP=34b
OR_A	0CC9H	EP=0c9
OR_IA	0C25H	EP=025
OR_IS	0E25H	EP=225

OR_S	0CCDH	EP=0cd
OR_SV	0D15H	EP=115
OR_V	0D11H	EP=111
OR.X_SV	0F15H	EP=315
OR.X_V	0F11H	EP=311
PARITY_SV	0DC6H	EP=1c6
PARITY.X_SV	0FC6H	EP=3c6
PATE	0D85H	EP=185
PATU	0D84H	EP=184
PBKPT	0DB9H	EP=1b9
PFORK	0C46H	EP=046
PFORK@	0C47H	EP=047
PICH	0D86H	EP=186
PLC.F_VM	0DDCH	EP=1dc
PLCH	0D87H	EP=187
PLC.T_S	0C96H	EP=096
PLC.T_V	0D4DH	EP=14d
PLC.T_VM	0DDDH	EP=1dd
PLC.T.X_V	0F4DH	EP=34d
POP.L_S	0DA7H	EP=1a7
POPR_A	0E10H	EP=210
POPR_A@	0E11H	EP=211
POP.W_A	0DA2H	EP=1a2
POP.W_S	0DA6H	EP=1a6
POWER.D_S	0D4AH	EP=14a
POWER.S_S	0D46H	EP=146
PROC_TRAP	0DFAH	EP=1fa
PROD.B_SV	0DD8H	EP=1d8
PROD.B.X_SV	0FD8H	EP=3d8
PROD.D_SV	0DD3H	EP=1d3
PROD.D.X_SV	0FD3H	EP=3d3
PROD.H_SV	0DD9H	EP=1d9
PROD.H.X_SV	0FD9H	EP=3d9
PROD.L_SV	0DDBH	EP=1db
PROD.L.X_SV	0FDBH	EP=3db
PROD.S_SV	0DD2H	EP=1d2
PROD.S.X_SV	0FD2H	EP=3d2
PROD.W_SV	0DDAH	EP=1da
PROD.W.X_SV	0FDAH	EP=3da
PSHEA	0C1AH	EP=01a
PSHEA@	0C1BH	EP=01b
PSH.L_S	0DA5H	EP=1a5
PSHR_A	0E12H	EP=212
PSHR_A@	0E13H	EP=213
PSH.W_A	0DA0H	EP=1a0

PSH.W_S	0DA4H	EP=1a4
PUT.L_S	0E6CH	EP=26c
PUT.L_S@	0E6DH	EP=26d
PUT.W_A	0E5CH	EP=25c
PUT.W_A@	0E5DH	EP=25d
RCV.L_S	0E66H	EP=266
RCV.L_S@	0E67H	EP=267
RCVR.L_S	0E1CH	EP=21c
RCVR.L_S@	0E1DH	EP=21d
RCVR.W_A	0E14H	EP=214
RCVR.W_A@	0E15H	EP=215
RCV.W_A	0E56H	EP=256
RCV.W_A@	0E57H	EP=257
RTN	0D92H	EP=192
RTNC	0D95H	EP=195
RTNQ	0D90H	EP=190
SHF_A	0CC1H	EP=0c1
SHF_IA	0C27H	EP=027
SHF_IS	0E27H	EP=227
SHF_JA	0C91H	EP=091
SHF_S	0CC5H	EP=0c5
SHF_SV	0D4CH	EP=14c
SHF_V	0D13H	EP=113
SHF_VSV	0D17H	EP=117
SHF.W_IS	0C33H	EP=033
SHF.W_S	0E91H	EP=291
SHF.X_SV	0F4CH	EP=34c
SHF.X_V	0F13H	EP=313
SHF.X_VSV	0F17H	EP=317
SIN.D_S	0D99H	EP=199
SIN.S_S	0D98H	EP=198
SND.L_S	0E6EH	EP=26e
SND.L_S@	0E6FH	EP=26f
SNDR.L_S	0E1AH	EP=21a
SNDR.L_S@	0E1BH	EP=21b
SNDR.W_A	0E18H	EP=218
SNDR.W_A@	0E19H	EP=219
SND.W_A	0E5EH	EP=25e
SND.W_A@	0E5FH	EP=25f
SPAWN	0E0AH	EP=20a
SPAWN@	0E0BH	EP=20b
SQRT.D_S	0DBBH	EP=1bb
SQRT.D_V	0EF5H	EP=2f5
SQRT.D.X_V	0EFDH	EP=2fd
SQRT.S_S	0DBAH	EP=1ba

SQRT.S_V	0EF4H	EP=2f4
SQRT.S.X_V	0EFCH	EP=2fc
ST.B_A	0C58H	EP=058
ST.B_A@	0C59H	EP=059
ST.B_S	0C68H	EP=068
ST.B_S@	0C69H	EP=069
ST.B_V	0C78H	EP=078
ST.B_V@	0C79H	EP=079
ST.B.X_V	0E78H	EP=278
ST.B.X_V@	0E79H	EP=279
STCMR	0E0EH	EP=20e
STCMR@	0E0FH	EP=20f
STE.B_S	0C48H	EP=048
STE.B_S@	0C49H	EP=049
STE.B.X_S	0E48H	EP=248
STE.B.X_S@	0E49H	EP=249
STE.H_S	0C4AH	EP=04a
STE.H_S@	0C4BH	EP=04b
STE.H.X_S	0E4AH	EP=24a
STE.H.X_S@	0E4BH	EP=24b
STE.L_S	0C4EH	EP=04e
STE.L_S@	0C4FH	EP=04f
STE.L.X_S	0E4EH	EP=24e
STE.L.X_S@	0E4FH	EP=24f
STE.W_S	0C4CH	EP=04c
STE.W_S@	0C4DH	EP=04d
STE.W.X_S	0E4CH	EP=24c
STE.W.X_S@	0E4DH	EP=24d
ST.H_A	0C5AH	EP=05a
ST.H_A@	0C5BH	EP=05b
ST.H_S	0C6AH	EP=06a
ST.H_S@	0C6BH	EP=06b
ST.H_V	0C7AH	EP=07a
ST.H_V@	0C7BH	EP=07b
ST.H.X_V	0E7AH	EP=27a
ST.H.X_V@	0E7BH	EP=27b
ST.L_S	0C6EH	EP=06e
ST.L_S@	0C6FH	EP=06f
ST.L_V	0C7EH	EP=07e
ST.L_V@	0C7FH	EP=07f
ST.L_VLS	0C1CH	EP=01c
ST.L_VLS@	0C1DH	EP=01d
ST.L.X_V	0E7EH	EP=27e
ST.L.X_V@	0E7FH	EP=27f
STOP	0D8EH	EP=18e

STVI.B_S	0DECH	EP=1ec
STVI.B_V	0DE8H	EP=1e8
STVI.B.X_S	0FECH	EP=3ec
STVI.B.X_V	0FE8H	EP=3e8
STVI.H_S	0DEDH	EP=1ed
STVI.H_V	0DE9H	EP=1e9
STVI.H.X_S	0FEDH	EP=3ed
STVI.H.X_V	0FE9H	EP=3e9
STVI.L_S	0DEFH	EP=1ef
STVI.L_V	0DEBH	EP=1eb
STVI.L.X_S	0FEFH	EP=3ef
STVI.L.X_V	0FEBH	EP=3eb
STVI.W_S	0DEEH	EP=1ee
STVI.W_V	0DEAH	EP=1ea
STVI.W.X_S	0FEEH	EP=3ee
STVI.W.X_V	0FEAH	EP=3ea
ST.W_A	0C5CH	EP=05c
ST.W_A@	0C5DH	EP=05d
ST.W_S	0C6CH	EP=06c
ST.W_S@	0C6DH	EP=06d
ST.W_V	0C7CH	EP=07c
ST.W_V@	0C7DH	EP=07d
ST.W.X_V	0E7CH	EP=27c
ST.W.X_V@	0E7DH	EP=27d
ST.X_VM	0C1EH	EP=01e
ST.X_VM@	0C1FH	EP=01f
SUB.B_S	0CECH	EP=0ec
SUB.B_SV	0D2CH	EP=12c
SUB.B_V	0D28H	EP=128
SUB.B.X_SV	0F2CH	EP=32c
SUB.B.X_V	0F28H	EP=328
SUB.D_S	0CD7H	EP=0d7
SUB.D_SV	0D1FH	EP=11f
SUB.D_V	0D1BH	EP=11b
SUB.D.X_SV	0F1FH	EP=31f
SUB.D.X_V	0F1BH	EP=31b
SUB.H_A	0CE8H	EP=0e8
SUB.H_IA	0C2AH	EP=02a
SUB.H_IS	0E2AH	EP=22a
SUB.H_JA	0CEAH	EP=0ea
SUB.H_S	0CEDH	EP=0ed
SUB.H_SV	0D2DH	EP=12d
SUB.H_V	0D29H	EP=129
SUB.H.X_SV	0F2DH	EP=32d
SUB.H.X_V	0F29H	EP=329

SUB.L_S	0CEFH	EP=0ef
SUB.L_SV	0D2FH	EP=12f
SUB.L_V	0D2BH	EP=12b
SUB.L.X_SV	0F2FH	EP=32f
SUB.L.X_V	0F2BH	EP=32b
SUBR.D_SV	0F01H	EP=301
SUBR.D.X_SV	0F05H	EP=305
SUBR.S_SV	0F00H	EP=300
SUBR.S.X_SV	0F04H	EP=304
SUB.S_IS	0E31H	EP=231
SUB.S_S	0CD6H	EP=0d6
SUB.S_SV	0D1EH	EP=11e
SUB.S_V	0D1AH	EP=11a
SUB.S.X_SV	0F1EH	EP=31e
SUB.S.X_V	0F1AH	EP=31a
SUB.W_A	0CE9H	EP=0e9
SUB.W_IA	0C2BH	EP=02b
SUB.W_IS	0E2BH	EP=22b
SUB.W_JA	0CEBH	EP=0eb
SUB.W_S	0CEEH	EP=0ee
SUB.W_SV	0D2EH	EP=12e
SUB.W_V	0D2AH	EP=12a
SUB.W.X_SV	0F2EH	EP=32e
SUB.W.X_V	0F2AH	EP=32a
SUM.B_SV	0DC0H	EP=1c0
SUM.B.X_SV	0FC0H	EP=3c0
SUM.D_SV	0DD1H	EP=1d1
SUM.D.X_SV	0FD1H	EP=3d1
SUM.H_SV	0DC1H	EP=1c1
SUM.H.X_SV	0FC1H	EP=3c1
SUM.L_SV	0DC3H	EP=1c3
SUM.L.X_SV	0FC3H	EP=3c3
SUM.S_SV	0DD0H	EP=1d0
SUM.S.X_SV	0FD0H	EP=3d0
SUM.W_SV	0DC2H	EP=1c2
SUM.W.X_SV	0FC2H	EP=3c2
SYSC	0C21H	EP=021
TAC.B	0C10H	EP=010
TAC.B@	0C11H	EP=011
TAN.D_S	0DBDH	EP=1bd
TAN.S_S	0DBCH	EP=1bc
TAS.B	0C18H	EP=018
TAS.B@	0C19H	EP=019
TC_TRAP	0DFCH	EP=1fc
TRAP	0C34H	EP=034

TR_TRAP	0DF9H	EP=1f9
TST	0E02H	EP=202
TST@	0E03H	EP=203
TSTVV	0DAFH	EP=1af
TZC_S	0C97H	EP=097
TZC_V	0D48H	EP=148
TZC.X_V	0F48H	EP=348
ULK	0E06H	EP=206
ULK@	0E07H	EP=207
VV_TRAP	0DFBH	EP=1fb
WFORK	0D93H	EP=193
XMTI	0DADH	EP=1ad
XOR_A	0CCAH	EP=0ca
XOR_IA	0C26H	EP=026
XOR_IS	0E26H	EP=226
XOR_S	0CCEH	EP=0ce
XOR_SV	0D16H	EP=116
XOR_V	0D12H	EP=112
XOR.X_SV	0F16H	EP=316
XOR.X_V	0F12H	EP=312
XPND.X_V	0F4AH	EP=34a

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix D
Wire Lists

To Be Determined

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix E
Reporting Problems

E.1 Overview

The *contact* utility is the recommended way to report minor hardware deficiencies and technical documentation problems to the Technical Assistance Center (TAC). This utility is an interactive tool that prompts the user for the information to properly file a problem report.

> **NOTE**
>
> The *contact* utility is not intended for requesting customer service for hardware failures. To restore your CONVEX equipment to operational status, faster service can be obtained by directly telephoning the TAC (refer to "Technical Assistance" in the preface).

To use the *contact* utility, there must be a phone connection to the TAC. A UNIX-to-UNIX Communication Protocols (UUCP) allows communication between UNIX systems by either dial-in or hard-wired communication lines. For more information, refer to *uucp(1)* or to the *info(1)* entry in the UNIX man pages.

The name and version number of the product involved is required. Use the *vers* command to ascertain the program or utility name and version. The syntax for the command is **vers** *filename*, where *filename* is the full pathname of the program. If the full pathname of the program is not known, enter **which** *program*. For more information, refer to the *vers(1)* and *which(1)* entries in the UNIX man pages.

E.2 Information Required to Report a Problem

The *contact* utility requires the following information:

1. The customer name, title, phone number, and corporate name
2. The hardware nomenclature, part number, and revision level, or the technical manual name, document number, and version

> **NOTE**
>
> Use *vers* and *which* to identify product name and version.

3. A short (one line) summary of the problem .
4. The more information provided, the more quickly the problem can be isolated and solved. At a minimum, include a detailed description of the problem (including page references, if applicable), the source code, and a stack backtrace whenever possible.

> **NOTE**
>
> See the *adb(1)* or *csd(1)* man pages for information on obtaining stack backtraces.

5. The priority of the problem, selected from a list of six levels
6. Instructions on how to reproduce the problem, including the command syntax used, any flags invoked, or anything else attempted to make the program run
7. Any other comments about the problem or files to be submitted

The *contact* user has a chance to review and edit the report prior to submitting it. If the user decides to delay submitting the report, the session can be aborted. The report is automatically saved in the user's top-level directory in a file named *dead.report.*

See the following figure for a sample *contact* session. User input is in bold lettering, and the system response is in monospace type.

Figure E–1, Sample *contact* Session

```
%contact (RETURN)
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
> Margaret Atwood, systems programmer, 814-4444, University r
> of Chicago (RETURN)
> (CTRL-D)

Enter the name of the product involved
> CONVEX UNIX Programmer's Manual, Part I (RETURN)

Enter the version number (in the form X.X or X.X.X.X) of the product
> Revision 4.0 (RETURN)

Enter a short (1 line) summary of the problem
> The finger command manual page lists nonexistent bug (RETURN)

Enter a detailed description of the problem (^D to terminate)
> The finger(1) man page says, under the BUGS section, that "Only the first
line of the .project file is printed."  Happily, this is not true! (RETURN)
> (CTRL-D)

Enter a problem priority, based on the following:
1) Critical    - work cannot proceed until the problem is resolved.
2) Serious     - work can proceed around the problem, with difficulty.
3) Necessary   - problem has to be fixed.
4) Annoying    - problem is bothersome.
5) Enhancement - requested enhancement.
6) Informative - for informational purposes only.
> 4 (RETURN)

Enter the instructions by which the problem may be reproduced (^D to terminate)
> a) put more than one line in .project (RETURN)
> b) read the man page for finger(1) (RETURN)
> (CTRL-D)

Enter any comments that are applicable (^D to terminate) (RETURN)
> (CTRL-D)

Do you have any suggestions or comments on the documentation that you
referenced when you were trying to resolve your problem (for example,
additions, corrections organization, accessibility)? (^D to terminate)
> The man page should be updated. (RETURN)
> (CTRL-D)

Are there any files that should be included in this report (yes | no)?
> no (RETURN)

Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
> 3 (RETURN)

Problem report submitted.
%
```

THIS PAGE INTENTIONALLY LEFT BLANK

Index

I

L

M

N

O

P

R

S

T

U

V

W

THIS PAGE INTENTIONALLY LEFT BLANK

CONVEX Troubleshooting Guide
(C201, C202, C210, C220)
Document No. 081-000930-201, First Edition, Rev. 1

Reader's Forum

You are invited to submit comments concerning the clarity and service of this manual. Constructive critical comments are most welcome, and will help us continue in our efforts to generate quality customer documentation. Please list the document page number with your questions and comments.

_____

_____

_____

_____

_____

_____

_____

_____

From:

Name_____ Title_____

Company_____ Date_____

Address and Phone No._____

FOR ADDITIONAL INFORMATION OR DOCUMENTATION:

Location	Phone Number
In Texas	(214)952-0200
Other continental locations	1(800)952-0379
Outside continental US	Contact local CONVEX office

Direct mail orders to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

(Fold Here First)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1046 RICHARDSON, TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

CUSTOMER SERVICE
CONVEX Computer Corp.
P.O. Box 833851
Richardson, TX 75083-3851

(Fold Here Second)

(Tape or Staple)