System Troubleshooting

HMM-114-B
CRAY J90 Series
Last Modified: April 1997

Figures

Tables

Record of Revision

April 1995

Original printing.

Revision A: October 1995

Revised to include information about the CRAY J932 mainframe and other miscellaneous changes.

Revision B: April 1997

Revised to include additional information on troubleshooting Y1 channel-related errors, diagnostic listings on the Web, and other miscellaneous changes.

Overview of this Document

This document describes basic hardware and software troubleshooting techniques for service personnel who have had CRAY J90 series training.

How to Use this Document

To use this document effectively, the troubleshooter should be familiar with the topics covered in the “Overview of this Document” section and then proceed to the “System-level Troubleshooting” section. After completing the system-level troubleshooting procedure, the troubleshooter must make some assumptions about which part of the system is failing (IOS, CPU, memory, peripheral, etc.) and then continue to the appropriate section of the document. This document is divided into five sections, which are described as follows:

1. Overview of Maintenance Environment - This section covers background information about how to begin troubleshooting your system.

2. System-level Troubleshooting - This section provides an overall approach to troubleshooting the system.

3. IOS Troubleshooting - This section covers the IOS troubleshooting procedure and the basics of IOS software troubleshooting.

4. Peripheral and Controller Troubleshooting - This section describes the tools available for troubleshooting peripheral problems.

5. CPUs and Memory - This section describes the tools available for troubleshooting CPU and memory problems.

NOTE: Power and cooling troubleshooting information is documented in *Power, Cooling, and Control*, Cray Research publication number HMM-100-A.

Related Publications

You need access to the following documents to use this document effectively:

- Refer to *Safety and ESD Guidelines*, Cray Research publication number HGM-016-A, for information about ESD and safety guidelines.
- Refer to the *CRAY J916 Service Manual Kit*, Cray Research part number HMK-101-0, for the entire set of hardware documentation for the CRAY J916 system.
- Refer to the *CRAY J90 Series System Programmer Reference*, Cray Research publication number CSM-0301-0A0, for information about configurations and hardware architecture.
- Refer to *CRAY IOS-V Messages*, Cray Research publication number SQ-2172 8.0.3, for descriptions of the IOS panic messages.
- Refer to *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3.2J, for descriptions of the IOS commands. The command descriptions in this manual are also available as online manual (man) pages.
- Refer to the *CRAY Y-MP, CRAY X-MP EA, and CRAY X-MP Computer Systems UNICOS Online Diagnostic Maintenance Manual*, Cray Research publication number SPM-1012 8.0, for descriptions of the UNICOS online diagnostic commands. The command descriptions in this manual are also available as online man pages.
- Refer to the *OLNET Online Diagnostic Network Communications Program Maintenance Manual for UNICOS*, Cray Research publication number SPM-1021, for information about the online network test OLNET.

- Refer to the *UNICOS Administrator Commands Reference Manual*, Cray Research publication number SR-2022 8.0, for descriptions of the UNICOS system administrator commands. The command descriptions in this manual are also available as online man pages.

- Refer to the *UNICOS Basic Administration Guide for CRAY J90 and CRAY EL Systems*, Cray Research publication number SG-2416 8.0.3.2.J, for information about basic UNICOS system administration.

ESD and Safety Guidelines

You must be familiar with the equipment and procedures necessary to prevent ESD from damaging the system. For more information regarding the control of ESD, refer to *Safety and ESD Guidelines*, Cray Research publication number HGM-016-A.

Replacing Defective Components

Refer to *Field Replacement Procedures*, Cray Research publication number HMM-079-B, if you need to replace a system component.

Submitting Corrections and Additions

This document will be updated periodically to incorporate new troubleshooting tools, procedures, and information that cannot be added at the time of this printing.

Please forward any corrections or additions to this document by using the reader comment form, by sending e-mail to *spt@cray.com*, or by using the feedback form that is provided at the Service Publications and Training Web site at `http://servinfo.cray.com`.

Overview of Maintenance Environment

This section provides an overview of all the troubleshooting tools that are supplied with your CRAY J90 series system.

System Console

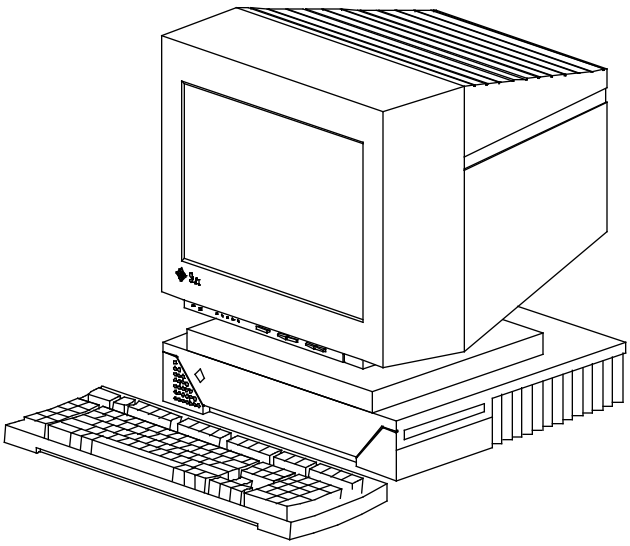
Cray Research uses a Sun Microsystems, Inc. SPARCstation® 5 workstation (refer to Figure 1) as the system console for CRAY J90 series systems. The system console provides a CD-ROM drive for loading system software. The system console also contains two SCSI disk drives to store the IOS kernel and configuration files, UNICOS kernel and configuration files, IOS and mainframe utilities and offline diagnostics, and other maintenance files as listed in the "Maintenance File Systems" section on page 11.

The system console connects to the master IOP (usually IOS 0) or a slave IOS. The `jcon` command establishes communication between the system console and the IOP as shown below, where *xxxx* represents your system serial number and *n* represents the IOS number.

```
snxxxx-ios0> jcon snxxxx-iosn
```

The `jcon` process automatically reconnects with the IOS if communication with the IOP is temporarily lost. Enter ~. and then Control-c (within 2 seconds) to exit the `jcon` process. Multiple `jcon` sessions running in multiple windows on the system console allows for concurrent connections to multiple IOSs; usually only one connection to the master IOP is needed.

Figure 1. Sun Microsystems, Inc. SPARCstation 5 Workstation



IOSNET

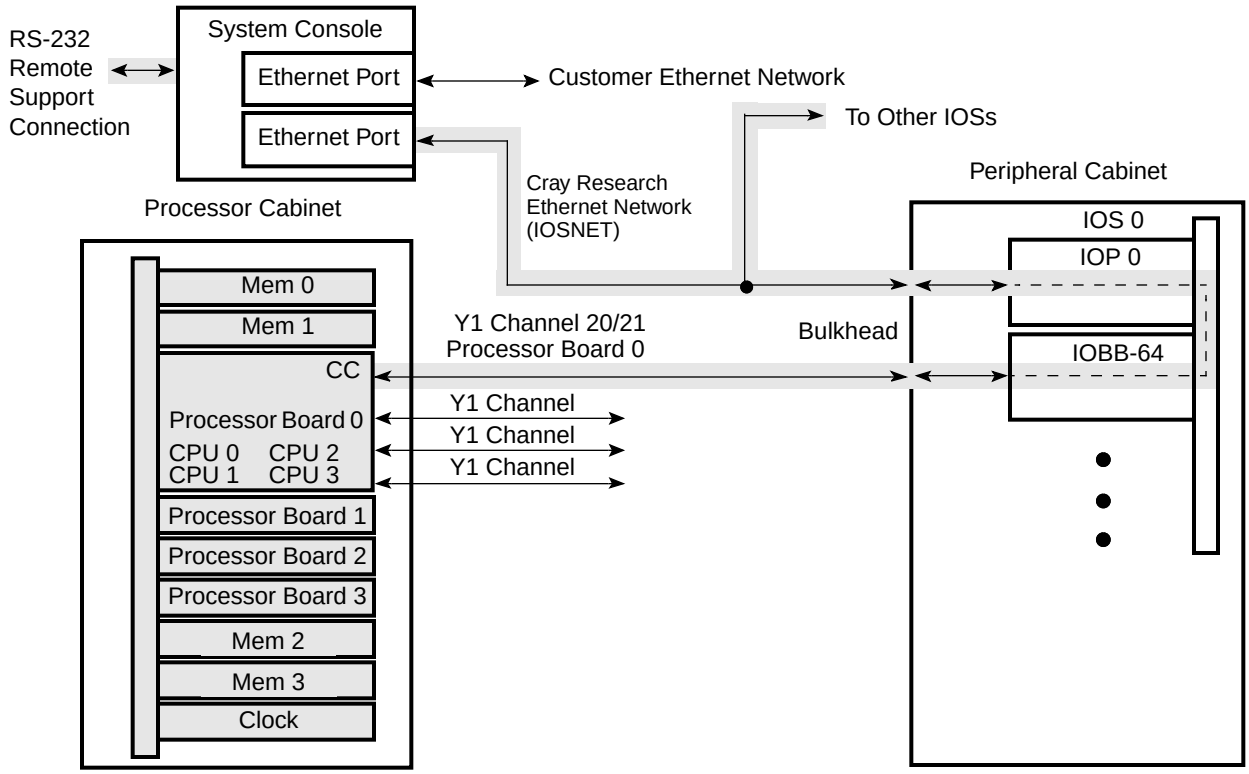
Figure 2 highlights the IOSNET. The system console private Ethernet connection enables communication with each I/O processor (IOP). This network is referred to as the IOSNET. **Connecting the IOSNET to the customer Ethernet network can cause the system to function unpredictably.** You can also access the system console by using the remote support equipment, which consists of the system console RS-232 port, modem, and optional NetBlazer® router.

NOTE: If the optional NetBlazer is used, it connects to the console through a third Ethernet board and not through the RS-232. The RS-232 is used only for modem support.

The IOSNET is connected to each IOP in the system. Each IOP connects to the mainframe CPUs with a Y1 channel. Input/output transfer control blocks (IOTCBs) are sent across the Y1 channel (normally channel 20/21) to the channel control (CC) application-specific integrated circuit (ASIC).

Processor board 0 and processor board 1 have extra backplane connections that enable them to communicate directly with the console bus (conbus) through the CC ASIC. Physical Y1 channels are associated with a specific backplane slot, but the logical Y1 channel can be accessed by each CPU.

Figure 2. System Console Connection



Each processor board contains two CC ASICs (CC0 and CC1). The CC ASICs can be configured as either two Y1 channels or as a HIPPI channel pair, except for CC0 on CPU 0 (channels 20/21), which must function as Y1 channels for proper CPU to IOS control (refer to Table 1).

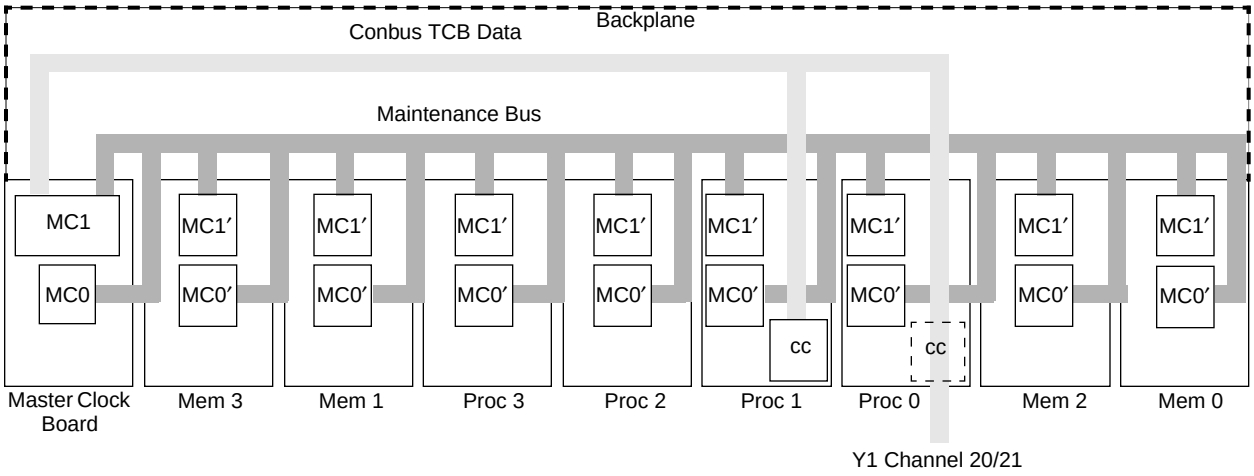
Table 1. CPUs and Associated I/O Channel Numbers

Processor Board	Y1 Channels		Y1 Channels		Y1 Channels		Y1 Channels	
	Input	Output	Input	Output	Input	Output	Input	Output
0	**20**	**21**	22	23	24	25	26	27
1	30	31	32	33	34	35	36	37
2	40	41	42	43	44	45	46	47
3	50	51	52	53	54	55	56	57
4	60	61	62	63	64	65	66	67
5	70	71	72	73	74	75	76	77
6	100	101	102	103	104	105	106	107
7	110	111	112	113	114	115	116	117

NOTE: All channel numbers listed are octal numbers.

The CC option on processor board 0, channel 20/21, sends the data and control signals to the MC1 ASIC on the master clock board (refer to Figure 3). The MC1 ASIC takes the TCB data and uses it to perform various maintenance functions such as Joint Test Action Group (JTAG) functions, internal scan, stop clock, and reset. The MC1 ASIC on the master clock board also distributes the maintenance signals to the MC1' ASICs on the other processor and memory boards in the system. The MC0 ASIC on the clock board distributes the maintenance clock signals to the MC0' ASICs on the processor boards and memory boards and to the clock board.

Figure 3. Internal Maintenance Channel ASICS



Maintenance File Systems

Table 2 lists some of the important IOS and UNICOS directories and files on the system console SCSI disk. When you are logged into the system console, each pathname is preceded by `/opt/ios/`*serialnum*`/`, where *serialnum* is the serial number of the system. When you are logged into an IOS, a prompt such as `sn9004-ios0>` signifies which IOS you are logged into. In this example it is IOS 0.

The system console `/opt/ios/`*serialnum*`/` directory (shown as ~ in Table 2) is Network File System (NFS™) mounted to root (`/`) on each IOS through the IOSNET. The files and directories displayed from the IOS console window in the root directory are the same files that reside in the `/opt/ios/`*serialnum*`/` directory on the system console.

Table 2. System Console Maintenance Files and Directories

Description	IOS Pathname
ACT scripts	~/diag/scripts/
ACT - escalated support menu	~/iosdiag/act_menu
ACT - diagnostic error information	~/adm/act_err.log
Boundary scan error file	~/adm/jbs.log
Disk maintenance commands	~/bin/dflawr, dflaww, dformat dslip, dstat, dsurf
IOS peripherals microcode identification	~/adm/mic_code.log
Disk flaw tables	~/flaw/*ios*/*device*
IOS kernel	~/ios/ios
IOS kernel symbol table	~/ios/ios.sym
IOS maintenance commands	~/bin/
IOS based diagnostics	~/iosdiag/
IOS configuration file	~/config
IOS error log	~/adm/syslog
IOS dumps	~/adm/dump*x*/Dmmddyy.*n*
IOS connection	/opt/home/crayadm/bin/jcon
Mainframe diagnostics	~/diag/ymp/
Mainframe dump (mfdump) configuration file (ASCII)	~/sys/mfdumpa.arg
Mainframe configuration files	~/sys/*.cfg
IOP Ethernet addresses	/etc/ethers
IOS strategies and drivers	~/dev/
UNICOS configuration file	~/sys/param

Automated Confidence Testing

Automated confidence testing (ACT) is a suite of shell scripts and diagnostic tests that detect and isolate hardware failures in CRAY J90 series computer systems; ACT provides two levels of system testing. The first level of testing, ACT power-up, is invoked automatically when the system is powered up and the IOS kernel is loaded. The power-up tests isolate defective system components and record or display error information. The second level of testing, the ACT menu system, provides a menu-driven interface to select and run specific diagnostics. Each level can be used by on-site, remote support, or escalated support service personnel to troubleshoot the system.

NOTE: If the file `/diag/scripts/frstload` has been renamed or removed, the power-up self-test will not run. However, the tests can still be accessed from the ACT main menu; ACT uses the `frstload.def` instead.

To successfully troubleshoot your CRAY J90 series system, you must be able to interpret the failure information that ACT displays on the system console. You must also be able to replace the defective component and retest the system using ACT, IOS based tests, or mainframe-based diagnostics to verify normal operation of the replaced component.

Refer to the *Automated Confidence Testing* document, Cray Research publication number HDM-110-A, for more information about ACT.

Mainframe Offline Diagnostics

The mainframe-based diagnostics are executed in the CPUs on one or more processor boards and can test the CPU functional units, central memory, I/O functions, and other system hardware. The mainframe diagnostics are configured and invoked automatically when you select them from the ACT menu system.

The CRAY J90 series mainframe offline diagnostics are modified versions of CRAY Y-MP style offline diagnostics. In order to successfully use the mainframe offline diagnostics, you must have a thorough knowledge of CRAY Y-MP diagnostics, and you must also have access to the diagnostic listings. You may access the diagnostic listings via the Service Publications and Training Web site at `http://servinfo.cray.com`.

The `offline` utility loads and configures the diagnostic tests that are stored on the system disk. The `offline` utility should be used only by escalated support personnel. Refer to “Mainframe Offline Diagnostics” on page 60 for more information about running mainframe offline diagnostics using the `offline` utility.

IOS Based Diagnostics and Utilities

Two types of IOS based tests are available: menu-driven tests and quick-look tests. The menu-driven confidence and comprehensive tests provide more extensive hardware testing than the quick-look tests and also provide an interface to select and run specific sections of each test. A confidence test should be used as a preventive maintenance check to test the basic operation of the hardware. A comprehensive test checks all circuitry in the hardware being tested and should be used when the hardware is operating incorrectly. The *IOS Based Diagnostics* document, Cray Research publication number HDM-099-0, describes the IOS-based tests.

The quick-look tests are scaled-down versions of the menu-driven tests that are used for automated confidence testing (ACT) to verify the system hardware during power-up. The quick-look tests can also be run from the `sn`*`xxxx`*`-IOSX>` prompt like the menu-driven tests but are designed primarily for ACT. The IOS quick-look tests are described in *Automated Confidence Testing*, Cray Research publication number, HDM-110-A.

UNICOS Online Diagnostics and Utilities

UNICOS online diagnostics and utilities run concurrently with UNICOS. Refer to the *CRAY Y-MP, CRAY X-MP EA, and CRAY X-MP Computer Systems UNICOS Online Diagnostic Maintenance Manual*, Cray Research publication number SPM-1012, for more information about UNICOS online diagnostics.

Remote Support

The remote support equipment consists of a modem and an optional NetBlazer router. Refer to the *Remote Support* document, publication number HMM-073-B, for more information about remote support.

System-level Troubleshooting

This section guides you through checking error messages, UNICOS and IOS error logs, and any system failure indicators. Gather as much information as possible about the failure before you perform any maintenance activities. If you run diagnostics or reset the system, you will destroy valuable failure information.

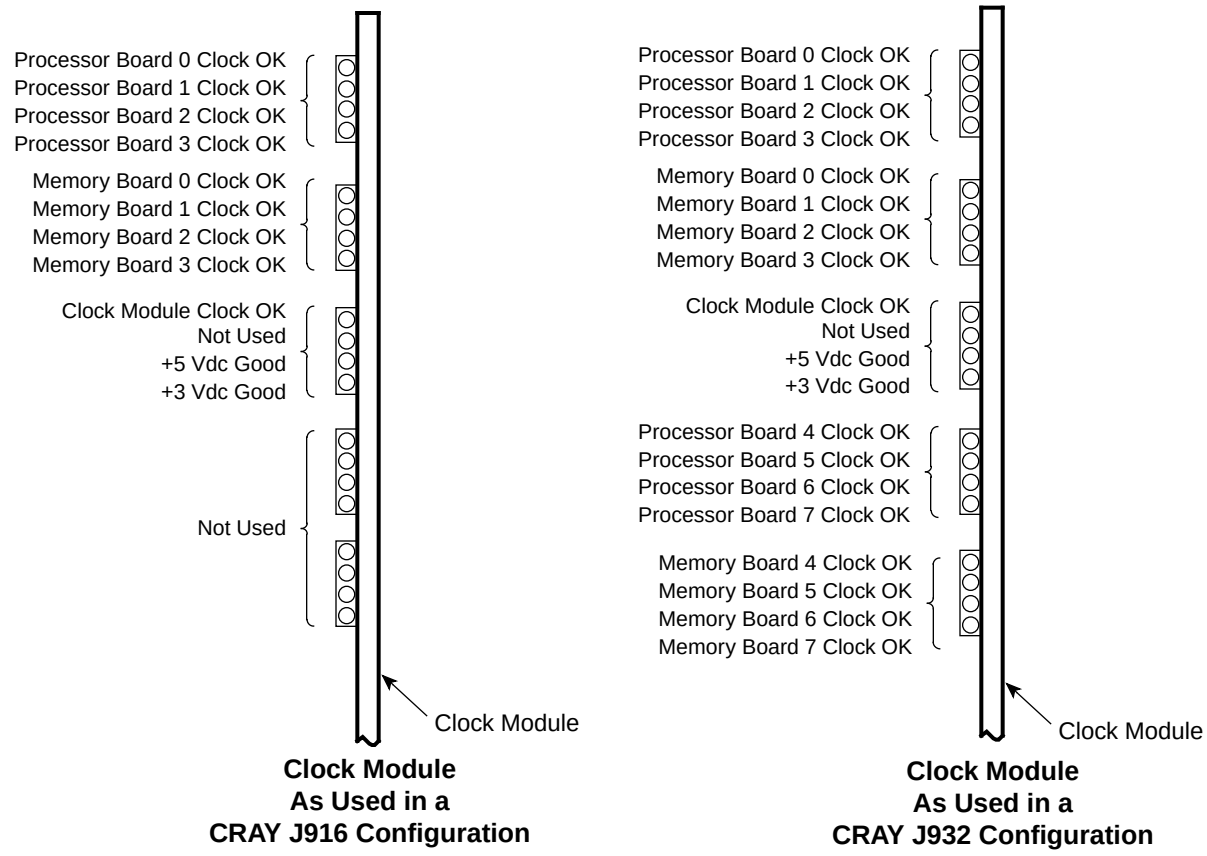
System Clock Module Troubleshooting

If the system is inoperable and you are unable to run any diagnostics, such as boundary scan or ACT, you may have a faulty clock module. The clock/scan module is located inside the mainframe cabinet. Refer to the *Field Replacement Procedures* document, publication number HMM-079-B, for information about removing the mainframe cabinet panels to access the clock module.

There are five banks of four green LEDS mounted to the clock module; refer to Figure 4. When these LEDs are illuminated, they indicate the clock status from each module and the clock voltage status. If the LED indicates a fault condition, (not illuminated), replace the appropriate module or run the boundary scan test, `jbs`, to test the backplane.

NOTE: If the CPU is not present, the associated LED will not illuminate.

Figure 4. Clock Module LED Status



System-level Troubleshooting Procedure

The system-level troubleshooting procedure in Table 3 attempts to identify the failing area of the system such as a processor or memory failure, an IOS or peripheral failure, or a power and cooling fault. It is a general approach to system failures. Based on your expertise with the system, you may use your own method to troubleshoot the system.

Table 3. System-level Troubleshooting Procedure

Procedures	Questions/Actions
Step 1. Determine the Status of the System The following procedure is a general approach to determining the status of the system.	
a. Ask any of the operators or system administration personnel whether anything has changed on the system recently.	Check to find out whether new software or hardware has been added to the system. After you have analyzed the failure data, try to determine whether the failure is related to the new modifications.
b. Check to see whether UNICOS and the IOS kernel are still running. Enter Control-a to toggle between the IOS console and the UNICOS console windows and verify that the IOS kernel and UNICOS are running. **NOTE:** You should verify that UNICOS is actually hung by trying to log into the system through the customer's normal network route (Ethernet, HIPPI, or FDDI). If it appears that UNICOS is still running but you cannot access it from the IOS console, you should quit the window using the window menu and then try to reestablish the connection.	The `jcon` process continuously attempts to reconnect to IOS 0. If `jcon` cannot connect to IOS 0, press the VME reset button on the central control unit (CCU). If the `BOOT>` prompt is not displayed in a few minutes, refer to the "IOS Troubleshooting Procedure" on page 34. If the IOS kernel is still running and UNICOS has crashed, refer to "Examine and Save Failure Information" as described in Step 2. If the IOS kernel is still running and UNICOS appears to be hung, use the IOS command `stat` to examine the state of all CPUs configured in the system. If UNICOS and the IOS kernel are still running, you may have had an intermittent failure. Refer to "Examine and Save Failure Information"as described in Step 2.
c. Open the front panel of the mainframe cabinet and check the CCU for power or cooling faults.	Record any failures indicated by the CCU. If power and cooling faults exist, refer to the *Power, Cooling, and Control* document, Cray Research publication number HMM-100-A. Check the environmental conditions in the computer room. The computer room temperature should range between 55 and 85 °F (13 and 29 °C); 70 °F (21 °C) is preferred.
d. Did the system power down automatically?	Refer to the *Power, Cooling, and Control* document, Cray Research publication number HMM-100-A.

Procedures	Questions/Actions
Step 2. Examine and Save Failure Information Be sure to save as much information as possible about the failure before you clear or reset the system. This information is very important for escalated support and repair personnel, especially when the failure is intermittent. Always save copies of the `/adm/syslog` and `/adm/mic_code.log` files after an IOS dump has been taken so that these files can be examined with the dump. The following procedures show you how to examine and capture error information from the system.	
a. Check the IOS 0 console window for IOS assertion panics, IOS processor panics, or UNICOS system panics. These should also be stored in `/adm/syslog`.	If IOS assertion panic messages are displayed, record the messages and capture an IOS dump as described on page 23 and then a UNICOS dump as described on page 24. Refer to page 40 for information about IOS panic messages.
b. Did the IOS 0 kernel remain active while UNICOS crashed?	Examine the IOS `/adm/syslog` file for error messages. Use the following three commands on each running IOS in the system: Use the IOS `errpt` command as described on page 28 to display system errors. Use the IOS `dstat` command as described on page 28 to display disk status. Use the IOS `systat` command as described on page 29 to display IOS activity.
c. If the IOS kernel and UNICOS are still running, you may have had an intermittent failure.	Check the IOS `/adm/syslog` file for error messages. Use the IOS `errpt` command as described on page 28, the UNICOS online hardware performance analyzer `olhpa` as described on page 27, or the UNICOS `errpt` command as described on page 28 to determine whether hardware errors exist. Use the IOS `dstat` command as described on page 28 to display disk status. Use the IOS `systat` command as described on page 29 to display IOS activity.

Procedures	Questions/Actions
Step 3. Reset and Clear the System The following paragraphs provide some tips for bringing up the system in order to continue troubleshooting. At this point you should have captured an IOS dump for each IOS and a UNICOS dump for escalated support personnel and examined the error information as described in the previous step "Examine and Save Failure Information".	
a. Reset the system. **Do not reset the system if UNICOS is running.** If UNICOS is running, use the UNICOS shutdown command before you proceed with this step to preserve as much user and system data as possible.	Open the front panel of the processor cabinet and press the CPU RESET and VME RESET buttons. **Always press CPU RESET first.** If a `jcon` session is currently running to IOS 0, then monitor that console window as IOS 0 is reset.
b. Can the system console connect to IOS 0?	If there is no current console connection to IOS 0, use the `jcon` command from the system console to connect to IOS 0. It may take a few minutes to reestablish the connection. If you cannot establish a connection to IOS 0, refer to "IOS Troubleshooting" on page 31.
c. Does the IOS PROM firmware load successfully and display the `BOOT>` prompt on the IOS 0 console window?	Refer to "IOS Troubleshooting"on page 31.
d. Enter `load` to load the IOS kernel. Does the IOS 0 kernel load and display the `snxxxx-ios0>` prompt?	Refer to "IOS Troubleshooting" on page 31.
e. Do the slave IOSs (if any) boot?	Refer to "IOS Troubleshooting" on page 31.
f. Enter `mc` to master clear the system. Is the `snxxxx-ios0>` prompt displayed?	It could be a defective IOBB, Y1 channel, or processor board 0, or configuration file problem. Has anything been reconfigured lately using the `j90install` tool or `jconfig`?

Procedures	Questions/Actions
Step 4. Run ACT After you have gathered all of the failure information and saved it, you should begin system-level troubleshooting using automated confidence testing (ACT). The ACT scripts run a series of diagnostics that test the system, starting with the most basic functions and progressing to more advanced functions. Depending on your situation, the type of failure information you have gathered, and the time you have to do troubleshooting, you may want to tailor your testing with the ACT menu. For example, option 4 from the comprehensive test menu reruns the power-up confidence tests. If the power-up tests pass, you should be able to boot UNICOS. If a failure occurs, begin at the basic test menu and proceed through each level of testing. Refer to the ACT man pages or *Automated Confidence Testing* document, Cray Research publication number HDM-110-A, for more information about the ACT system.	
a. Enter **`act_menu`** at the `sn`*`xxxx`*`-ios0>` prompt to invoke the ACT menu system. The ACT menu system provides three levels of testing: basic, intermediate, and comprehensive.	If you have no idea what the failure is, start by running all the basic tests (option 1). If all tests pass, proceed to the intermediate testing menu and continue testing. **Each test must successfully complete before you run the next level of testing.** If you have analyzed the failure information and recognized a failure related to a specific portion of the system, you should use `act_menu` to test that specific area.
b. Does the ACT menu system detect a failure?	When a failure occurs, the ACT system displays a field replaceable unit (FRU) list, starting with the most probable cause of failure. Refer to the ACT man pages or *Automated Confidence Testing*, Cray Research publication number HDM-110-A, for more information about ACT system failure information. If no FRU is listed, ACT was unable to decode the failure. Stop running ACT and begin testing with diagnostics. Customers who support their own systems need to contact technical support.

Procedures	Questions/Actions
Step 5. Boot UNICOS If the ACT menu system tests have run successfully, you should then attempt to boot UNICOS.	
a. Enter `boot` at the `sn`*`xxxx`*`-ios0>` prompt. **NOTE:** Some sites may have customized boot scripts. Be sure to contact site administration or operators before booting the system.	Monitor the console when UNICOS boots. Note the status information when the system comes up in single-user mode. Check for any warning or error messages. These error messages are written to the `/adm/syslog` file. Verify that all configured CPUs are running; check the available memory information, controller information, etc.
b. If UNICOS boots to single-user mode, verify the root file system by entering `/etc/fsck /dev/dsk/roota`. **NOTE:** Some sites may be using a different `inode` for the `root` file system; contact the site administrator if you are unsure. If the `root` file system is ok, verify the `usr` file system by entering `/etc/fsck /dev/dsk/usra`. **NOTE:** Some sites may be using a different `inode` for the `root` file system; contact the site administrator if you are unsure.	Observe the output from `fsck`. If file system problems are found, `fsck` will attempt repairs; answer "yes" to any questions about `fsck` repair problems.
c. If the `/usr` file system is ok, enter `/etc/init 2` to invoke multi-user mode.	Carefully observe the monitor when UNICOS boots and enters multi-user mode.

Procedures	Questions/Actions

Step 6. System Specific Troubleshooting

At this point you should have taken system dumps, gathered as much information as possible from the error logs and error report formatting tools, and run the ACT menu system to verify the hardware.

If UNICOS has successfully booted in multi-user mode, the failure may have been intermittent, in which case the error information and dumps will be very useful for escalated support personnel. You could also run UNICOS mainframe online diagnostics as described on page 59.

If you think that the failure is related to the IOS, CPU, central memory, peripherals or controllers, or power and control, continue to the appropriate section of this document or refer to other documents listed in “Related Publications”on page 6 of this document for further guidance in these areas.

Step 7. Contact Escalated Support

If the failure cannot be isolated, you should contact escalated support for assistance.

Dumping the IOS(s) Using iosdump

If your CRAY J90 series system experiences an IOS assertion panic, IOS processor panic, or a UNICOS system panic, you must perform an IOS and mainframe dump to save the data in memory and use the results to determine the cause of the panic. Panic messages are issued as a result of critical, unrecoverable system failures. These error messages are logged in the IOS `/adm/syslog` file and are assigned a message number.

When either an IOS assertion panic or an IOS processor fault panic occurs, the firmware on the IOS automatically captures a dump of that IOS and saves it to the system console disk in the IOS file `/adm/dump`*x*`/D`*mmddyy*`.`*n*. IOS dumps are typically 9 Mbytes for each IOP. In cases where an apparent hang or unusual system behavior has occurred, you must manually perform an IOS dump.

The `iosdump` command saves all of the memory from the IOP and selected areas from the I/O buffer board (IOBB) memory to the system console disk.

NOTE: You can invoke the `iosdump` command at the `BOOT>` prompt or the `snxxxx-ios0>` prompt. If your IOS is locked up, refer to "IOS Troubleshooting" on page 31.

The format of the `iosdump` command is as follows:

`iosdump [-n` *filename*`]` **`[-s`** *iobbsize*`]`

The arguments are optional. The default area dumped during an IOS dump is not a complete dump of IOBB memory; however, it usually contains enough information for problem analysis. To obtain a complete dump, use the `-s` option and enter the memory size in Kbytes of the IOBB. For example; for an I/O subsystem configured with an IOBB-64, you would enter `-s 16384`. Dumps that are automatically initiated by the IOS will be of the default size; you cannot control this. However, if an automatic dump is taken and the IOS stops at the `BOOT>` prompt, you can then initiate another valid dump if you must capture the entire IOBB contents. If the IOS kernel has reloaded, the IOBB and IOP contents will have been overwritten, so dumps taken after a reload are invalid.

By default, the IOS generates a filename for the contents of the dump. The following example displays the output of the `iosdump` command:

```
sn9004-ios0>iosdump -s 16384

Saving IOS dump to /adm/dump0/D011895.0...
Dump file size, 1MB, 2MB, 3MB, 4MB, 5MB, 6MB, 7MB, 8MB
IOS0 dump completed
sn9004-ios0>
```

Dumping a Slave IOS

To dump a slave IOS, use the system console `jcon` command as described on page 8 to connect to the slave IOS. Then perform the iosdump procedures as described on page 23.

Examining an IOS Dump

Continue with system-level troubleshooting procedures before you examine the dumps. System dumps are very useful for intermittent problems or software errors. Dump analysis requires a thorough knowledge of the system operations. If a solid hardware failure exists, you can isolate the system component more quickly using the automated confidence testing (ACT) scripts.

After you have completed some basic hardware troubleshooting procedures and have attempted to reboot the system, you can examine the dump contents by using the IOS `crash` command as described on page 42.

Dumping the Mainframe Using mfdump

If your CRAY J90 series system experiences an IOS assertion panic, IOS processor panic, or a UNICOS system panic, you must perform a UNICOS dump to save the data in memory and use the results to determine the cause of the panic. Panic messages are issued as a result of critical, unrecoverable system failures. These error messages are logged in the IOS `/adm/syslog` file and are assigned a message number.

To perform a system dump:

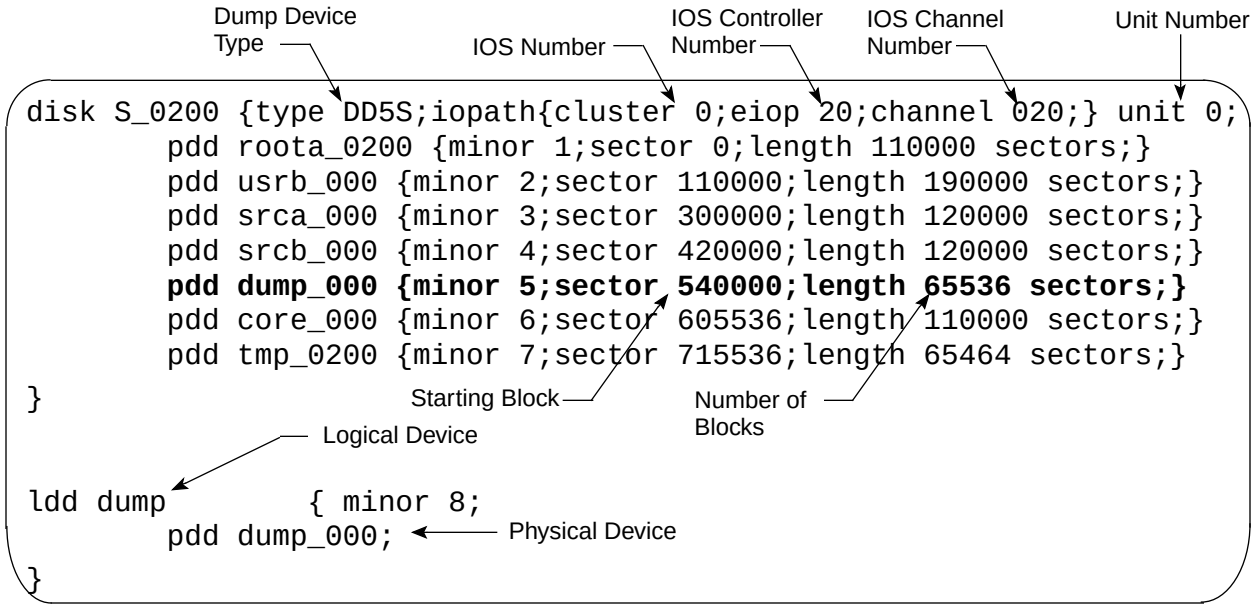
1. If the IOS is not running, reset the VME.
2. Perform an IOS dump using the `iosdump` command if a dump has not already been done.
3. After the IOS dump has been taken, reload the IOS.
4. Perform the mainframe dump using `mfdump`. (Refer to the following subsections for more information about using `mfdump`.)

Refer to *CRAY IOS-V Messages*, Cray Research publication number SQ-2172 8.0.3.2J, for descriptions of the IOS panic messages.

Refer to *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3.2J, or the online manual (man) pages for descriptions of the IOS commands.

Configuring mfdump

Before you can capture a UNICOS dump for the first time, there are several tasks you must perform. You must be sure that you have allocated enough disk space to receive the dump. The minimum size of the dump device should be a little larger than the amount of memory you actually want to examine in order to provide an additional 512 words for the dump header plus additional space for ublocks, which vary by system. You should start with a minimum of 50,000 blocks (sectors) for the dump size. The following example shows the dump entry in the physical and logical device portion of the UNICOS file system section of the IOS `/sys/param` file, which is normally configured during installation.



The name of the disk partition is usually `/dev/dsk/dump`.

NOTE: Before you can use the dump device, the system administrator must use the `mkdmp` command to initialize the dump device. This is normally done when UNICOS is installed.

Usually, the initialization of the dump partition and copying of the dump are accomplished in the `/etc/coredd` start-up script when you initiate multi-user mode. The following paragraphs provide tips for configuring `mfdump` correctly.

- The `mfdump` command is not supported from the `BOOT>` prompt. You must have the IOS loaded before running `mfdump`.

- When you run `mfdump` for the first time, check the `/sys/mfdumpa.arg` (ASCII version) argument file. Its contents are described in the man page for `mfdump`; an example follows. During installation, the `/sys/mfdumpa.arg` argument file is configured for the system. You can use the `mfdump -c` command line option to display the current settings for the dump configuration file.

```
CPUS=4
MEM=64
range1=0-8000000
range2=0-0
range3=0-0
range4=0-0
regdump=yes
sysreg=yes
ublocks=yes
IOS#=0
channel#=16
disktype=DD5S
controller=20
unit=0
start=540000
length=65536
```

- If there is a system dump in the dump partition that has not been copied, use the `-f` command line option to force the dump. This will overwrite the previously uncopied dump.

Running mfdump

The following snap shows an example of using the `mfdump` command. You may specify the reason for the dump with the `-r` option.

```
snxxxx-ios0>mfdump -f -v -r System panic on an ORE in user code
.
```

Verifying That You Have Captured a UNICOS Dump

After a UNICOS dump is captured using `mfdump`, it is copied off the dump device and into a file system when multi-user mode is initiated. Check for the following output on the console when UNICOS is booted in multi-user mode. For more information about this process, check the `/etc/brc` script and refer to the UNICOS man page on `mkdmp`.

```
/core: file system opened
/core: super block fname core, fpack core_000
/core: clean exit for clean file system
/etc/mount: warning <core> mounted as </mnt>
01446 blocks - NOT copied
coredd: Copying system dump into /mnt/08230838.
Sysdump copy completed
/etc/umount: /mnt unmounted successfully

                    :
```

NOTE: Many systems use the `/tmp` file system to store dumps instead of the `/core` file system that is shown in the previous example.

Providing the UNICOS Dump to Escalated Support

System dumps are very useful for intermittent problems or software errors. If there is a solid hardware failure, the system component can be isolated more quickly using the automated confidence testing (ACT) scripts.

UNICOS dumps are very useful for hardware and software product support personnel, and you should forward them to the appropriate person for analysis. The `unicos` and crash files that are created along with the dump file become the core file system and should also be forwarded for analysis.

Using the UNICOS olhpa Command

If UNICOS is running, you can use the online hardware performance analyzer (`olhpa`) to format and display the contents of the UNICOS `/usr/adm/errfile` file. Enter `olhpa` with any of the following command line options at the UNICOS prompt. Entering `olhpa` with no option is equivalent to entering `olhpa -dmti`.

Option	Function
`-d`	To display disk errors
`-m`	To display memory errors
`-t`	To display tape errors
`-i`	To display IOS communication driver errors
`-q`	To display a quick summary of all errors
`-l`	To display a long version or error message

Refer to the online man page or the *CRAY Y-MP, CRAY X-MP EA, and CRAY X-MP Computer Systems UNICOS Online Diagnostic Maintenance Manual*, Cray Research publication number SPM-1012 8.0, for a detailed description of `olhpa`.

Using the UNICOS errpt Command

If UNICOS is running, you can use the error report formatter (`errpt`) to display the contents of the UNICOS `/usr/adm/errfile`. The `errpt` command processes data collected by the error-logging mechanism `errdemon` and generates a report of that data.

Refer to the online man page or the *UNICOS Administrator Commands Reference Manual*, Cray Research publication number SR-2022 8.0.2J, for a detailed description of `errpt`.

Using the IOS errpt Command

You can use the `errpt` command to display memory resident error status. This information is lost if you reboot the IOS kernel.

Refer to the *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3, or the online manual (man) pages for a description of the `errpt` command.

The `errpt` command displays a buffer that contains all the error packets that this IOS sent to UNICOS from the time the IOS was loaded until the `iosdump` command was executed. These error packets are buffered in UNICOS in the `errfile` that is displayed by the UNICOS command `errpt`.

Using the IOS dstat Command

The `dstat` command executed without command line options briefly summarizes disk activity since the IOS was booted. Optionally, the name of a specific disk may be included on the command line to request more disk-specific information. If any disk errors are logged in syslog, you can use `dstat` to check the failing device by specifying the appropriate option. Refer to the *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3.2J, or the online man page for a description of the `dstat` command.

This command accepts the following options:

Option	Function
`-b`	Indicates a buffered IPI disk drive
`-s`	Indicates a SCSI disk drive
`-c`	Specifies controller number (0 to F)
`-d`	Specifies disk (0 to F)

You can also specify a device by its device designation as shown in Figure 5.

Figure 5. IOS dstat Command Output

```
sn9004-ios0>dstat s00
Driver Strategy Numbers:
        Number of driver calls made  : 920
        Number of chained calls made : 265
        Times requests were suspended: 579


        SCSI Adapter 0 [s0]: 2 drives found
            Drive  Total Reads    Total Writes  Outstanding
              0          424           442             0
            Drive  Total Reads    Total Writes  Outstanding
              1            0            54             0

        SCSI Adapter 1 [s1]: 2 drives found
            Drive  Total Reads    Total Writes  Outstanding
              0            0             0             0
            Drive  Total Reads    Total Writes  Outstanding
              1            0             0             0

        SCSI Adapter 0, Drive 0, Serial Number : 710552
            Data Cylinders:  2737  Heads       : 21
            Sectors/Track :    14  Sector size : 4096
            4K Blocks available for data : 781000
            Device does support SCSI-2 tags
            Device queue depth is 8 requests
```

Using the IOS systat Command

The `systat` command displays the current status of various parts of the IOS and network status (refer to Figure 6). The display includes IOBB buffer pool numbers, queued IOBB packets, open file descriptors, amount of IOS memory available, and slave IOS information, if applicable.

The IOS network-status table portion of the `systat` display provides information regarding your master and slave IOSs, if applicable. The `state` field of this table describes the state of the appropriate IOS. This portion of the `systat` display is included only when `systat` is executed on the master IOS.

Refer to the *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3 or the online man page, for more information about the `systat` command.

Figure 6. IOS systat Command Output

```
sn9004-ios0>systat
Buffer Pool Status:
       buf size:      128 bytes, tot:  32, free:  32, used: 0    (reserved)
       buf size:     1024 bytes, tot:   4, free:   4, used: 13
       buf size:     4096 bytes, tot:  33, free:  33, used: 11
       buf size:    49152 bytes, tot:   2, free:   2, used: 0    (reserved)
       buf size:   131072 bytes, tot: 123, free: 123, used: 208

        getblks: 232          relblks: 232          waiting : 0
        waited : 0         exact fits: 232          big fits: 0

Transfers To Mainframe: 0 queued (5120 max; 0 queued IDX pkts)
Open file descriptors : 12

IOS network status:
IOS  state     network status   load sent   last response received
___  ________  _______________  _________  _________________________

0    RUNNING       Active          Yes      THU JAN 19 07:42:46 1995
1    RUNNING       Active          Yes      THU JAN 19 07:42:30 1995
```

IOS Troubleshooting

This section guides you in troubleshooting IOS failures. An IOS consists of the input/output processor (IOP), I/O buffer board (IOBB), peripherals, and controllers. If you suspect that your failure is related to a peripheral or controller, you may want to verify the proper operation of the IOP and IOBB before you troubleshoot peripherals or controllers.

Examining Failure Information

You can locate IOS error information by any of the following methods:

- Examine the IOS `/adm/syslog` file from the system console for error messages.
- Use the IOS `errpt` command, as described on page 28, to display system errors.
- Use the IOS `enstat` command to display Ethernet controller status.
- Use the IOS `dstat` command, as described on page 28, to display disk activity.
- Use the IOS `systat` command, as described on page 29, to display IOS activity.
- Use the UNICOS `errpt` command to display the contents of `/usr/adm/errfile` as described on page 28.
- Use the UNICOS online hardware performance analyzer `olhpa`, as described on page 27, to determine whether the hardware errors are related to an IOS.

IOP Status LEDs

You can obtain more information about the state of the IOS(s) by looking at the eight red status LEDs on each IOP. You must open the front door on the I/O cabinet and pull out the VME chassis drawer in order to see the LEDs. Refer to the numbered examples in Figure 7 and the following text for an explanation of the IOS LED indications.

NOTE: If no LEDs are illuminated or only one LED is illuminated steadily, this usually indicates that the board has faulted and the IOP is in a hung condition. An IOS reset from the CCU is required.

Example 1: When the IOS kernel is uncompressing programmable read-only memory (PROM mode), LEDs 0 and 1 are illuminated. This normally occurs very quickly.

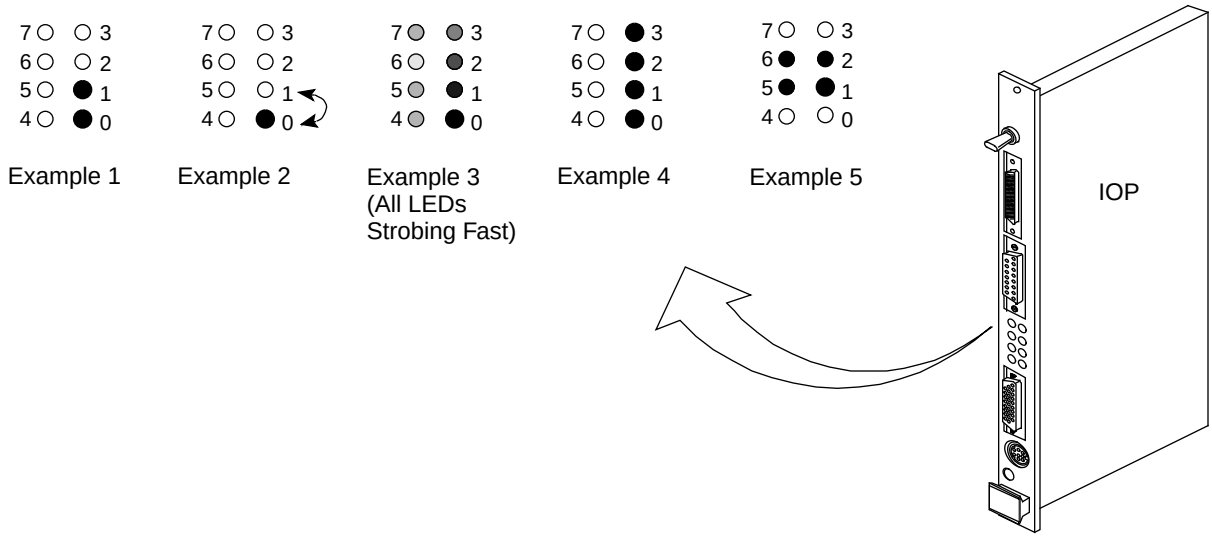
Example 2: When LEDs 0 and 1 are strobing, the IOP is operating in PROM mode. The `BOOT>` prompt should be displayed in the IOS console window. Use the `jcon` command or `rlogin` command to reconnect to the IOS if the `BOOT>` prompt is not displayed.

Example 3: When the IOS kernel has been loaded and is running, all LEDs on the front panel sequentially strobe and the `snxxxx-ios0>` prompt is displayed in the IOS console window.

Example 4: When LEDs 0 through 3 are illuminated steadily, there may be a Cray Research private Ethernet network (IOSNET) problem such as a loose or missing cable terminator, or loss of the NFS connection to the system console.

Example 5: When LEDs 1, 2, 5, and 6 are illuminated steadily, the control system cable is not connected to the power rail assembly. If this cable is connected on both the VME cage and the power rail assembly, there could be a problem with the power rail assembly or the VME.

Figure 7. IOS LED Indications



IOS Load Sequence

In the CRAY J90 series IOS-V, IOS 0 receives the `load` command first and early in the load process (even before any drivers are loaded in); the master IOS (IOS 0) initiates the load sequence in the other IOSs when it determines that additional IOSs are present. All IOSs then load in parallel. Pay attention to the system console during the IOS load sequence; warning and error messages will display if an IOS fails to load properly.

After an IOS is reset, the IOSs come up in PROM mode. The IOSs idle in PROM mode, and all slave IOSs communicate to IOS 0 (every 5 seconds) that they are functioning and are running in PROM mode. This continues indefinitely until you enter the `load` command on IOS 0.

When IOS 0 receives the `load` command, it loads the IOS kernel. The IOS kernel initializes its hardware, signals the system console to access its root file system, NFS mounts its root file system, and reads the `/config` file.

Once IOS 0 has read the `/config` file, it has the list of all its slave IOSs. IOS 0 signals its network monitor daemon that it is initialized to the point that the slave IOSs can begin to load. Remember that the slave IOSs are still in PROM mode and are still sending a status signal to the master every 5 seconds, which indicates that they are i n PROM mode and can be loaded. The network monitor can now respond to this status with the load command for each slave IOS. Each slave IOS receives a load command from IOS 0 within 5 seconds, which begins the load process in parallel.

As each slave IOS loads, it sends its output (device and information, etc.) to IOS 0, which buffers it without displaying it to the system console. When a slave IOS completes its load sequence, IOS 0 displays the buffered output at once so the output from one IOS load sequence is not intermixed with the load sequence from another IOS. An exception occurs when a slave IOS fails to load properly, which results in a notice or warning message that the system console displays immediately.

IOS Troubleshooting Procedure

This subsection describes the procedures for troubleshooting IOP and IOBB failures. Peripheral and controller troubleshooting is discussed briefly in this subsection and is explained in more detail in the next section "Troubleshooting Peripherals and Controllers," which begins on page 50.

The IOP runs the IOS kernel, which carries out I/O functions for the controllers, processes external interrupts, and executes peripheral driver routines. The IOBB-64 buffers data between the VME IOS and central memory. The buffer memory of the IOBB is allocated into a buffer pool (as shown by using the `systat` command, which is described on page 29) when the IOS kernel is initialized. The Y1 channel connects the IOBB-64 directly to a Y1 channel controller (CC) on the processor board. Typically, IOS 0 is the master IOP (MIOP), which initiates the maintenance functions and communicates with the system console.

Table 4 describes the "IOS Troubleshooting Procedure."

Table 4. IOS Troubleshooting Procedure

Procedures	Questions/Actions
Resetting the IOS(s)s If you press the VME RESET button on the processor cabinet each IOS resets. After a reset, the IOS(s) should return to PROM mode (the `BOOT>` prompt) within 2 minutes without corrupting IOP or IOBB memory. You can also reset each IOP individually if you remove the top cover from the VME chassis. You should note that UNICOS cannot recover from an IOS being reset. Resetting an IOS requires that UNICOS be rebooted. You should then take an IOS dump as described on page 23 and a UNICOS dump as described on page 24, if you have not already done so.	
a. Check the CCU for I/O cabinet faults.	Refer to the *Power, Cooling, and Control* document, Cray Research publication number HMM-100-A.
b. Press the VME RESET button on the CCU.	IOS 0 should return to PROM mode. The `BOOT>` prompt should be displayed in a few minutes.
c. Establish the connection between the system console and IOS 0. Use the `jcon` command or `rlogin` command from the system console to connect to IOS 0. It may take a few minutes to reestablish the connection.	Can you connect to IOS 0? If you cannot establish a connection to IOS 0, the most probable cause of failure is IOS 0, the Ethernet connection from the system console to the IOS 0, the system console Ethernet board, or an Ethernet addressing problem. Has anything been reconfigured lately with the `j90install` tool?
	Is the `BOOT>` prompt displayed? If you have connected to IOS 0 and the `BOOT>` prompt is not being displayed after a reset, the most probable cause of failure is the IOP for IOS 0.
d. Enter `load` at the `BOOT>` prompt to load the IOS kernel.	Does the IOS kernel panic, hang, or fail to load?
	If the IOS hangs or fails to load, ensure that the IOS `/config` file is present and readable (refer to page 39). Ensure that the IOP Ethernet address, strategy, and driver entries are correct for each IOS in the IOS `/config` file.

Procedures	Questions/Actions
d. Enter `load` at the `BOOT>` prompt to load the IOS kernel (continued).	Check the IOS `/dev` directory and make sure that all the drivers and strategies configured in the `/config` file are present. Try copying these into `/tmp` and watch for any read errors that indicate a corrupt file.
	Perform the previous action with the `/ios/ios` and `/ios/ios.sym` binary files. Examine the file sizes. Device driver files should range from 5 Kbytes to 100 Kbytes. If a device driver file is much larger than 100 Kbytes, data may have been inadvertently written over the driver file.
	Check the driver file creation or modification dates using the system console `ls -l` command. Any files with incorrect time stamps can be identified.
	Were any IOS panic messages displayed? Refer to “Interpreting IOS Error Messages” on page 40.
	Are any controller errors displayed or are any controllers not found?
	All controllers on the IOS VMEbus are capable of interfering with the operation of the others. If device errors are displayed, check for errors in the `/adm/syslog` file. Also consider removing the nonsupported controllers from the VMEbus. As controllers are identified by the IOS software, console messages and `/adm/mic_codex` file entries will be generated (`x` indicates the IOS number). After all IOSs have loaded, you can execute the IOS command `whatmic` to concatenate all of the `/adm/mic_codex` files into one file, `/adm/mic_code.log`. The `whatmic` command also sends this log file to the console for viewing. Use this file to verify that all installed peripherals were recognized by the IOS software during the load sequence.

Procedures	Questions/Actions
d. Enter `load` at the `BOOT>` prompt to load the IOS kernel (continued).	As an alternative, an IOS can be reduced to the IOP, IOBB, and the suspected faulty controller in adjacent slots, but make sure the slot that the suspected faulty controller now occupies does not have any interface connections attached to the rear of the VME backplane.
e. Isolate the failing component of the system by reducing the configuration. The configuration can be reduced in hardware, software, or both.	**Software**. During the IOS load sequence, device strategies and drivers are loaded in accordance with entries in the IOS `/config` file. Problems with any of these files (or the `/ios/ios` binary) can prevent the IOS kernel from loading. All other strategies and drivers can be commented out of the `/config` file. As long as the statement that defines an IOS's Ethernet address remains in the `/config` file, then the IOS kernel can be loaded into the IOS. If the failing IOS now loads successfully, the strategies and their corresponding driver(s) can be reinstated in the `/config` file, one at a time, until the bad strategy and driver combination is identified. Refer to "IOS Software Overview" on page 38 for more information.
	Hardware. The minimum hardware configuration that will support the IOS kernel is the IOBB and the IOP. Remove all controller boards from the backplane and try to load the IOS. If successful, reinstall the controllers one at a time, starting from the left, until the failing board is identified. If unsuccessful, replace the IOBB and then the IOP. Remember to enter the Ethernet address of the new IOP using `j90install` tool. If the IOS kernel does not load after you have swapped the IOBB and the IOP, check the cabling and jumpers. It is recommended that you remove the strategy and driver entries that correlate to "removed controller board" so that the software will not confuse your troubleshooting with its error/timeout messages. However, a strategy or driver must remain in `/config` if there is at least one controller of that type remaining in the system.

IOS Software Overview

This section provides an overview of the IOS software components. The `j90install` tool maintains the software on the IOS. The `j90install` tool is a windows-based application that runs on the system console. You can use this tool to load, reconfigure, and upgrade the IOS software and UNICOS software components.

Primary Software Components

The following list describes the primary components of the IOS kernel, strategies, and drivers:

- The IOS kernel (`/ios/ios`) passes packets and parses and executes commands. The IOS commands are for running diagnostics, booting the IOS and UNICOS, and invoking other utility functions. The symbol table file (`/ios/ios.sym`) of the IOS kernel must correspond to the `/ios/ios` file in order for the IOS kernel to load properly.

  IOS commands allow users to run diagnostics, load and boot the IOS kernel and UNICOS, display registers, and perform IOS and UNICOS memory dumps.

  UNICOS I/O requests are transferred to the IOS using packets. A packet is a data structure (up to 69 decimal words long and aligned on 128-word boundaries) that defines an I/O request sent across the command channel portion of the Y1 channel. Each packet contains a packet type, I/O request type (read or write), a mainframe memory address to write to or read from, I/O transfer length, and I/O completion status.

- Strategies are device independent. Strategies set up data transfers (UNICOS central memory to devices). Strategies also validate I/O requests and allocate memory in the IOBB. There is **one** strategy per **device type**; for example, tape, disk, etc.

- Drivers are binary files that contain **device-specific** logic. The device driver returns to the strategy when an I/O function completes.

  The device driver layer contains the device-specific code used to access and control devices. The device driver layer is responsible for transferring data to or from the IOBB and devices. Upon completion of the transfer, status information is sent to the strategy layer. The strategy layer then sends a response packet to UNICOS.

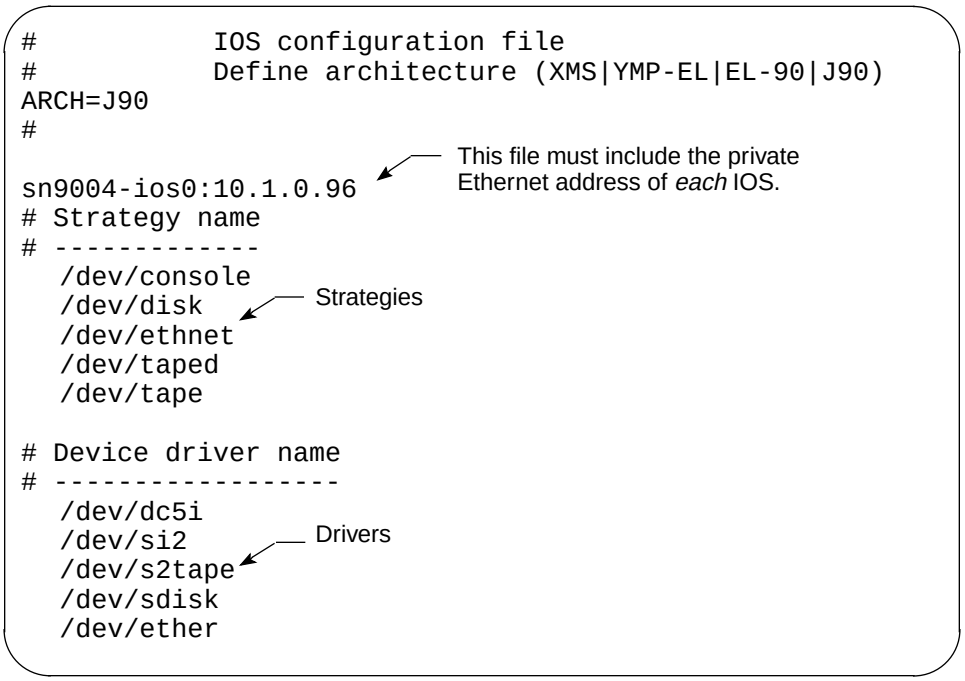
IOS /config File

The IOS `/config` file, shown in Figure 8, lists the strategies, drivers, and IOP Ethernet addresses that are loaded into IOP memory when the IOS(s) are loaded.

If your system has more than one IOS, it is helpful for troubleshooting efforts to create a secondary configuration file that loads only IOS 0. This enables you to quickly reset the entire system and run mainframe offline diagnostics from a reduced configuration. Alternatively, you can use the `-q` option on the IOS `load` command, which will load only the IOS kernel into the IOS where this command is executed; no other IOSs will be loaded.

System configuration files maintain the mainframe hardware configuration for CRAY J90 series systems. Refer to “System Configuration Utility jconfig” on page 73 for information about the mainframe hardware configuration files.

Figure 8. IOS /config File



IOS Strategies and Drivers

Table 5 lists the IOS strategies and drivers that the VME IOS supports.

NOTE: Strategies should always be loaded first in the `/config` file.

Table 5. IOS Strategies and Drivers Matrix

Strategies	Description
/dev/console	Console terminal driver. Required only for IOS 0 so that it can communicate with the UNICOS console driver
/dev/disk	Disk device strategy for all disk drives
/dev/ethnet	Ethernet network interface strategy
/dev/fdnet	FDDI network interface strategy
/dev/taped	UNICOS tape daemon strategy
/dev/tape	Strategy for all non-tape daemon tape interfaces
Drivers	**Description**
/dev/si2	SCSI interface driver for SI-3 controller; required for any SCSI disk or tape attached to an SI-3 controller; must be listed before s2tape or sdisk
/dev/sdisk	SCSI disk device driver for SI-3 attached disks
/dev/s2tape	SCSI tape driver for SI-3 attached tape devices and for all IOP attached tape devices (DAT)
/dev/dc5i	Buffered IPI (DD-5I) disk driver
/dev/ether	Ethernet network interface driver
/dev/fddi	FDDI network interface driver
/dev/atmv	ATM network interface driver

Interpreting IOS Error Messages

There are four types of IOS error messages: `PANIC`, `WARNING`, `NOTICE`, and `INFORMATION`. The messages appear on the system in the following format:

`ios`*# [message number] message type date time module line # routine : Message [message]*

```
sn9004-ios0: [IOSV-3006] ASSERTION PANIC: 02/07/95 14:36:01 e_task.c line 699
e_inhdr_compl: Data read from CM corrupted in IOBB. Take iosdump -s 4096 and mfdump.
Sccsid: @(#)e_task.c    1.3 e_task.c 1.3 94/11/07 16:26:25
        Kernel Version: Cray Research Incorporated IOS-V Kernel 1.3 95/01/30 15:35:49
        IOS Version 1.3
```

Refer to *CRAY IOS-V Messages*, Cray Research publication number SQ-2172, for a description of each IOS panic message.

Panic Messages

Panic messages are issued as a result of critical, unrecoverable system failures. These error messages are logged in the `/adm/syslog` file and are assigned a message number.

There are three types of panics:

- IOS assertion panic. An IOS assertion panic indicates that the IOS software has encountered a catastrophic problem. When this occurs, the console displays a message, similar to the previous example.

- IOS processor fault panic. An IOS processor fault panic indicates some type of unexpected hardware condition. When this occurs, the console displays a message, similar to the following one, that indicates the cause of the problem (*n* is the IOS number), which is followed by a register dump:

  `IOS`*n*: *error message text*
  register dump

- UNICOS panic. A UNICOS panic occurs when the UNICOS operating system encounters a catastrophic problem. When this occurs, the console displays a message, similar to the following one, that indicates the cause of the problem (*n* is the IOS number):

  `IOS`*n* `NOTICE:` *date time* `UNICOS PANIC:` *error message text*

Warning Messages

Warning messages are issued when some aspect of the system is not functioning properly. They may be the result of a hardware component failure, an unavailable resource, or some other serious condition that does not immediately cause the system to stop functioning. Warning messages are logged in the `/adm/syslog` file and are assigned a message number.

Notice Messages

Notice messages are issued as advisory messages that a problem might exist or a condition occurred but is not necessarily serious. These messages are logged in the `/adm/syslog` file but are not assigned a message number.

Information Messages

Information messages are issued as advisory messages that a problem might exist or a condition occurred but is not necessarily serious. Information messages are not logged in the `/adm/syslog` file and are not assigned a message number.

Using Crash

This section provides a brief overview of the `crash` command and some examples of how you can use it to gather more information about a failure. If you are not very familiar with the internal operations of both IOS hardware and software, you should contact escalated support for assistance before using `crash`.

Refer to *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3, or the online man page for a description of the IOS `crash` command.

The IOS `crash` command is an interactive program that formats and displays an IOS memory image (dump file) to the IOS console window.

NOTE: You cannot use `crash` to examine a running IOS. It is used only on dump files generated by the `iosdump` command. However, the `iosdump` command can be executed on a running IOS without affecting that IOS. The resulting dump file can then be examined using `crash`.

Prior to the UNICOS 8.0.4.1 release, the `crash` command could be executed only at the IOS prompt. Starting with UNICOS 8.0.4.1, the `crash` command has been transferred to the SPARC® console platform. As of UNICOS 8.0.4.1, the `crash` command can be executed only from the system console prompt. The `crash` command functions the same from either platform.

To examine an IOS dump file, enter `crash` and the name of the dump file, as shown below. To exit the `crash` command, enter q. The `crash` command requires addresses to be in hexadecimal numbers regardless of any prefixes. If necessary, `crash` converts an address to a file offset. Be sure to note the size of the dump to verify that you have captured a complete dump.

The following screen display shows an example of the `crash` command that was used with releases prior to the UNICOS 8.0.4.1 release.

```
sn9004-ios0>crash D011895.0
        namelist start: 512       size: 61821
         NVRAM   start: 62464     size: 2048
        iop mem  start: 64512     size: 8388608
        iobb mem start: 8453120   size: 409600
crash>
```

The following screen display shows an example of the `crash` command that was used with the UNICOS 8.0.4.1 release and subsequent releases.

```
sn9004>crash D011895.0
        namelist start: 512       size: 61821
         NVRAM   start: 62464     size: 2048
        iop mem  start: 64512     size: 8388608
        iobb mem start: 8453120   size: 409600
crash>
```

Useful Crash Command Options

The following `crash` command examples can be useful when you are gathering more information about a failure. For a complete list of all of the `crash` command options, refer to the *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3, or the online man page. The following `crash` command options are entered at the `crash>` prompt.

status

The `status` command lists the full panic messages, IOS number, and the version of the IOS kernel running at the time of the panic, along with the date and time the dump occurred, PROM version, server IP address, master IOP name and IP address, NFS root directory, and the panic string. Depending on the type of panic, you may also see a fault address and stack traceback entries in the status output. Use the fault address with the VME memory map, as described on page 46. This information is very helpful for escalated support personnel.

```
crash> status

Version: Cray Research Incorporated IOS-V Kernel 1.3 95/01/31 16:35:00

Reload: 0, panic: 0, powerup: 0, powerup_load: 0, qload: 0
Namelist size   : 61821 bytes
Trace Table Addr: 0x261680
Panic Time Stamp: TUE JAN 31 18:20:40 1995

Boot Time Stamp : UNICOS not booted yet
Load  Time Stamp: 31/01/95  18:14:21
Load File       : /ios/ios
Prom Version    : Cray Research IOS-V PROM 1.3 95/01/25 12:58:19 Part# #90422000
Kernel Version  : Cray Research Incorporated IOS-V Kernel 1.3 95/01/31 16:53:00
IOS Version     : 1.3
Server Name     : iroc-private
Server IP Addr  : 172.16.103.1
Master Name     : sn5612-ios0
Master IP Addr  : 172.16.103.2
NFS Root        : /opt/ios/5612
User Name       : crayadm
Panic String    :

sn5612-ios0: [IOS-V] PROCESSOR FAULT PANIC: TUE JAN 31 18:20:40 1995

Exception at interrupt level:

Exception: Memory Address Not Aligned
program counter:            0x001bf6cc
next program counter:       0x001bf6d0
processor status register:  0x904009c7
Exception frame at 0x299fd0

Kernel Version: Cray Research Incorporated IOS-V Kernel 1.3 95/01/31 16:53:00
IOS Version: 1.3
```

sysbuf

The `sysbuf` command displays the last entries that were sent to the `/adm/syslog` file. The system failure may have prevented the last message from being written to the `/adm/syslog` file, in which case you would not see the failure message in the `/adm/syslog` file.

ttybuf

The `ttybuf` command displays the contents of the IOS kernel's tty buffer. These messages may not have been written to the system console at the time of the panic.

Namelist

Use `nm` (namelist) to identify the executable routine or data area of a given address.

help

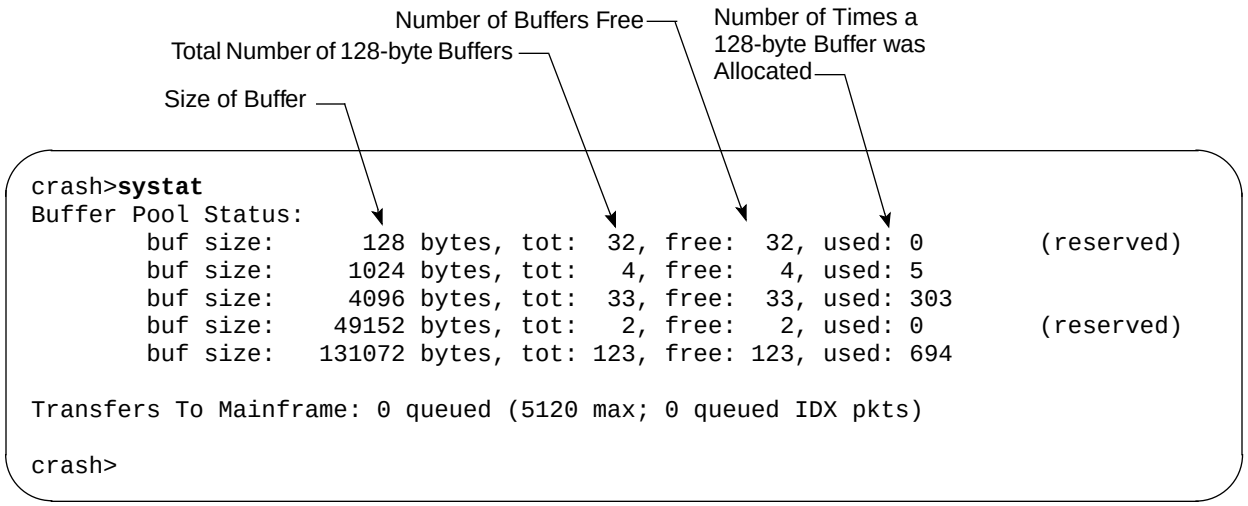
The `help` command lists all commands available at the `crash>` prompt.

errpt

The `errpt` command displays a buffer that contains all the error packets that this IOS sent to UNICOS from the time the IOS was loaded until the `iosdump` command was executed. These error packets are buffered in UNICOS in the `errfile` that is displayed by the UNICOS command `errpt`.

systat

The `systat` command displays the state of the IOBB buffer pool and IOBB transfer queue at the time of the IOS panic. You should verify that the system did not run out of buffer resources.




```
crash>systat
Buffer Pool Status:
        buf size:      128 bytes, tot:  32, free:  32, used: 0        (reserved)
        buf size:     1024 bytes, tot:   4, free:   4, used: 5
        buf size:     4096 bytes, tot:  33, free:  33, used: 303
        buf size:    49152 bytes, tot:   2, free:   2, used: 0        (reserved)
        buf size:   131072 bytes, tot: 123, free: 123, used: 694

Transfers To Mainframe: 0 queued (5120 max; 0 queued IDX pkts)

crash>
```

loadmap

The `loadmap` command lists each strategy, driver, or command that is run with the load address and size. The loadmap table has a finite set of entries. It is possible that not all of the strategies and drivers are listed in the output. Always disregard any command listed in the loadmap output. The `jobs` command in `crash` more accurately reflects which user commands were run before the execution of `iosdump`.

jobs

Use the `jobs` command to examine the last 16 user commands run on the IOS. Verify that a user did not inadvertently master clear the system or run an offline diagnostic that caused the system to crash. The `jobs` command can also be used from the IOS prompt.

```
crash> jobs
TID BG  NOHUP STATE    COMMAND
0   NO  CLR  DONE

1   NO  CLR  DONE     <old: cd cd>
2   NO  CLR  DONE     <old: ls>
3   NO  CLR  DONE     <old: cd cd>
4   NO  CLR  DONE     <old: ls>
5   NO  CLR  DONE     <old: cd>
6   NO  CLR  DONE     <old: cd cd>
7   NO  CLR  DONE     <old: ls>
8   NO  CLR  ACTIVE    iosdump -s iosdump
9   NO  CLR  DONE     <old: ls -l ls>
10  NO  CLR  DONE     <old: ls ls>
11  NO  CLR  DONE     <old: cd cd>
12  NO  CLR  DONE     <old: ls>
13  NO  CLR  DONE     <old: more more>
14  NO  CLR  DONE     <old: ls>
15  NO  CLR  DONE     <old: more more>
16  NO  CLR  DONE     <old: ls>
```

Options Not Supported

The following `crash` command options are not supported for CRAY J90 series systems that include the Themis™ VME IOS controller (IOS-V): `das esdi`, `ireq`, `pertec`, `stb PC A6_register`, `istat`, `itrace`, `s3560`, `si1`, and `treq`.

VME Memory Map

Depending on the type of panic, you will also see a fault address and stack traceback entries in the status output. Use the fault address with the VME memory map as shown below to see where the IOP was executing when the failure occurred. The following list describes the supported devices and their interrupt and memory map locations.

- The A16 memory map locations are physical addresses. When the processor attempts to access one of these boards on the VMEbus, it uses a virtual address that is displayed in the dump as the fault address. All A16 addresses are in the range 0x2FF0000 to 0x2FFFFFF. For example, the IOBB is at address 0x2FF1040.

  A16 Memory Map:

0x1040 to 0x1042	IOBB-64 control and status
0x6000 to 0x6200	FI-2 Controller 0
0x6200 to 0x6400	FI-2 Controller 1
0x7000 to 0x77ff	SI-3 Controller 0
0x7800 to 0x7fff	SI-3 Controller 1
0x8000 to 0x87ff	SI-3 Controller 2
0x8800 to 0x8fff	SI-3 Controller 3
0xC000 to 0xC0FF	DC-5I Controller 0
0xC100 to 0xC1FF	DC-5I Controller 1
0xC200 to 0xC2FF	DC-5I Controller 2
0xC300 to 0xC3FF	DC-5I Controller 3

- A24 addresses are in the virtual address range 0x2000000 to 0x2FFFFFF. For example, EI-1, controller 0 is at address 0x2E00000.

  A24 Memory Map:

0xE00000 to 0xE3FFFF	EI-1 Controller 0
0xE40000 to 0xE7FFFF	EI-1 Controller 1
0xE80000 to 0xEBFFFF	EI-1 Controller 2
0xEC0000 to 0xEFFFFF	EI-1 Controller 3A32

- A32 virtual addresses are the same as the physical address.

  A32 Memory Map:

0x04000000 to 0x04FFFFFF	IOBB-64 memory
0x05000000 to 0x057FFFFF	IOP DRAM

DRAM is accessed by two methods: A cached view of DRAM accessed from address 0 to 0x7FFFFF; an uncached view of DRAM is accessed from address 0x5000000 to 0x57FFFFF.

IOP Interrupt Vectors

When a UNICOS system panic occurs, the panic message may contain the IOP interrupt vector for a device, at the time the panic occurred. Table 6 shows the IOP interrupt vectors.

Table 6. IOP Interrupt Vectors

Vector	Device	Vector	Device
20	IOP SCSI device	83	SI-3 controller 3
22	IOP Ethernet (IOSNET)	100	IOBB-64
26	System clock	101	FI-2 controller 0
28	SCC (RS-232 port)	102	FI-2 controller 1
30	Auxiliary clock	105	EI-1 controller 0
70	DC-5I controller 0	106	EI-1 controller 1
71	DC-5I controller 1	107	EI-1 controller 2
72	DC-5I controller 2	108	EI-1 controller 3
73	DC-5I controller 3	176	AT-1 controller 0
80	SI-3 controller 0	177	AT-1 controller 1
81	SI-3 controller 1	178	AT-1 controller 0
82	SI-3 controller 2		

Y1 Channel Errors

Troubleshooting Y1 channel errors can be difficult because they can be caused by a failure of the Y1 cables, an IOBB, a Y1 paddle card, or the processor module. To assist you in troubleshooting a Y1 channel error, review the contents of the IOS `/adm/syslog` file for Y1 channel error messages. Y1 channel errors may also be listed by executing the `errpt` command if the IOS has not been rebooted.

You can test the Y1 channels by running the following diagnostics in order: `bb1test, bb2test, cc1test, cc2test`. Each test tests a deeper level of functionality. These diagnostics are run at the IOS operating system prompt or through the ACT basic menu. Before you run any of these tests, you must shut down the UNICOS operating system. Also, because each IOS has its own Y1 channel, you must log into the IOS that has the suspected Y1 channel failure.

You should begin troubleshooting Y1 channel errors by swapping Y1 cables in the IOS that is reporting the channel error. If you swap cables, be sure to note which cables were swapped. Also be aware that the cable failure could be intermittent; therefore, the swap may not immediately indicate the faulty cable. For isolation purposes, you should also move the IOBB from the IOS that is reporting the failure to another IOS.

If these actions fail to resolve the Y1 channel problem in that path, the next step would be to replace the IOBB and move the processor module.

Peripheral and Controller Troubleshooting

This section provides troubleshooting techniques you may use when you suspect that you have a peripheral or controller failure. If you are not sure which controller in an IOS is causing a failure, remove all controllers that are not supported by Cray Research from that IOS VMEbus.

Running IOS Based Tests and Utilities

You can use the IOS based tests and utilities to target a suspected peripheral or controller. Table 7 lists the IOS based tests and utilities and the area of the system for which they can be used. Refer to the *IOS Based Diagnostics* document, Cray Research publication number HDM-099-0, or the online man page for detailed descriptions of each IOS based diagnostic and utility.

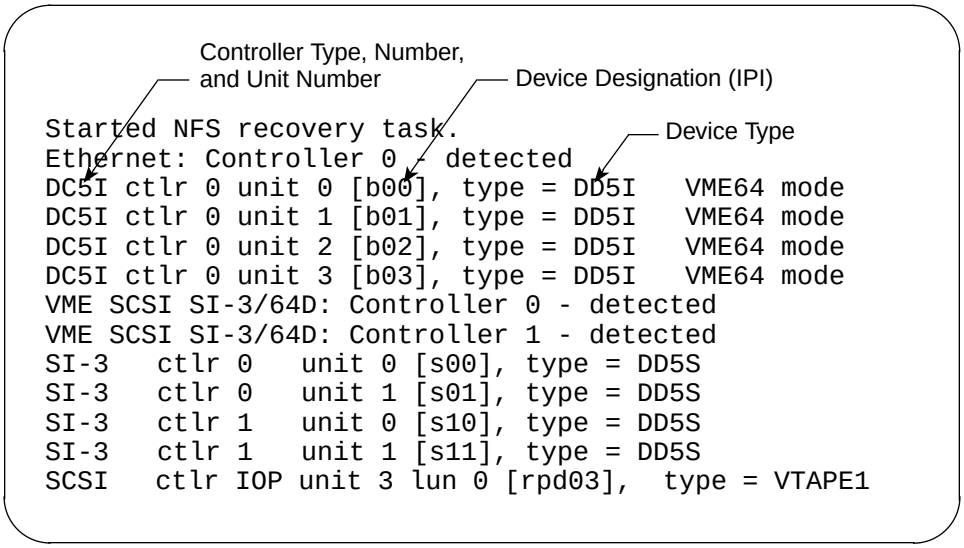
Table 7. Peripheral and Controller Tests and Utilities

Test or Utility	Peripheral or Controller
act_menu	ACT menu system can be used to test a variety of peripherals and controllers
bb1test	Checks IOBB functions
bb2test	Checks the related interfaces between the IOBB and the selected disk drive
cc1test	Checks the related interfaces between the IOP, IOBB, and central memory using data channel I/O
cc2test	Checks the related interfaces that control central memory (CM) to (IOBB) to CM data transfers
dd5itest	Checks DD-5I disk drives and DC-5I controllers
dd5stest	Checks any Cray Research-supported SCSI disk drive, the SI-3 controller, and the related interfaces between the disk drive and the IOS
enstat	Displays Ethernet controller statistics; does not include information about the private Ethernet console connection
mm1test	Checks IOP RAM and IOP cache memory
nettest	Checks communication on the Ethernet network and FDDI token ring network
tp1test	Checks the selected Cray Research-supported tape device, the IOBB, and the related interfaces between the tape device and the IOS; the IOBB must be functional for this test to be effective

Disk Drives

CRAY J90 series systems support two types of disk drive systems: those that use the buffered intelligent peripheral interface (IPI) controller, and those that use the SCSI controller.

You can examine the `/config` file for your system (refer to page 39 for information about the `IOS/config` file) to determine which disk drives are configured on each IOS. Save the IOS kernel load information as shown below. It displays the device designations and controller information, which are very helpful when you are troubleshooting. The `/adm/mic_code.log file`, which is displayed by the `whatmic` command, also contains helpful information about the disk configuration.



Disk Drive and Disk Controller Tests

Disk drives and disk controllers are typically tested by using the ACT menu system or the appropriate IOS based offline diagnostic (refer to Table 7).

Disk Error Information

You can locate disk error information by using any of the following methods:

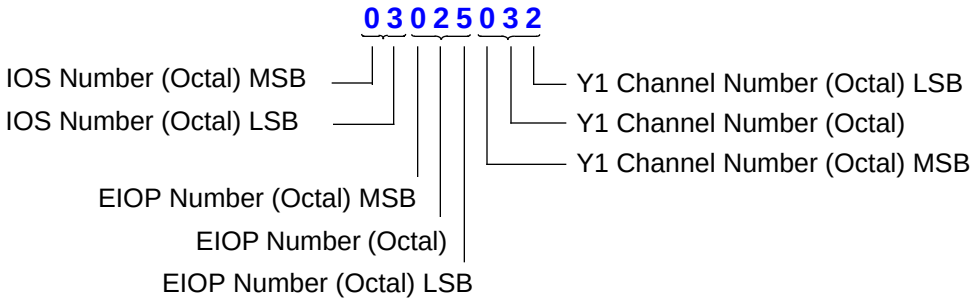
- Use the UNICOS `olhpa` command (refer to page 27)
- Use the UNICOS `errpt` command (refer to page 28)
- Use the IOS `dstat` command (refer to page 28)
- Examine the IOS `/adm/syslog` file

Error messages for disks can be difficult to decode because of the format that is used for the disk channel information. The following example of a system console error message explains how to decode disk channel information.

The following error message was displayed on the CRAY J916 console:

```
06:07:20 uts/c1/io/pdd.c-15: WARNING
disk 03025032: unit 0: request
time-out on path 03025032
```

The system crashed as a result of a request timeout in IOS 3. After running diagnostics, taking an IOS dump, and bringing up UNICOS, it was determined that Y1 channel 32 on SCSI controller 1 of IOP 1 in IOS 3 was the cause of the error.



Y1 Channel Number: Directly relates to the octal Y1 CPU channel port number.

EIOP Number: Must be converted to decimal to relate to the actual (UNICOS) representation of the EIOP number. (Refer to EIOP conversion table below.)

IOS Number: Must be converted to decimal to relate to the actual (UNICOS) representation of the IOS number.

EIOP Conversion for SCSI-based Disks			
024(o) =	020(d) =	First SI-3 in IOS;	Port 0
025(o) =	021(d) =	First SI-3 in IOS;	Port 1
026(o) =	022(d) =	Second SI-3 in IOS;	Port 0
027(o) =	023(d) =	Second SI-3 in IOS;	Port 1
030(o) =	024(d) =	Third SI-3 in IOS;	Port 0
031(o) =	025(d) =	Third SI-3 in IOS;	Port 1

Valid SCSI disk drive unit numbers are:
S00, S01, S20, S21, S30, S31, S40, S41

03025032 = IOS 3, logical SCSI disk controller 1, processor module 1, Y1 channel 32

Disk Maintenance

UNICOS is run on CRAY J90 series systems with the assumption that all disks have flaw-free media. This means that all flaw hiding, remapping, and data correction must occur at the IOS device and controller level. This also means that UNICOS does not maintain a UNICOS spares map. Because UNICOS does not recognize flaws, all flaw management must be performed at the IOS kernel level.

Flaw handling on disk drives is performed through a combination of sector slipping and track remapping. Each track contains a spare sector that can accommodate a defective sector. Spare tracks are set aside in the upper cylinders to accommodate track remapping in cases where slipping is not possible. If all spare tracks are in use, the drive must be replaced. All flaws are saved in the growth error table stored on the drive. The IOS kernel commands that perform flaw management include `dformat`, `dsurf`, `dslip`, `dflawr`, and `dflaww`.

- `dformat` - This command formats the entire drive.
- `dsurf` - The `dsurf` command performs surface analysis on a disk and provides a means to pattern-test and to flaw a drive.
- `dslip` - This utility attempts to read the specified sector up to 100 times while checking for errors. If errors are detected, the sector is slipped or the track is remapped. If no errors are detected by `dslip`, the user is informed and prompted as to whether slipping or remapping should proceed.
- `dflawr` - This utility reads the flaw table from the specified drive and writes an ASCII file copy to `/opt/ios/`*serialnum*`/flaw/`*ios#*`/`*dev.flw* on the system console disk drive. Use this command after formatting or flawing a drive in order to keep an emergency backup copy of the flaw table.
- `dflaww` - This utility reads the flaw table from the IOS disk and writes it to a system disk. The flaw table file used by `dflaww` is created with the `dflawr` command. This command is useful in emergencies in which the flaw table that resides on the disk itself has been corrupted and you want to recover the flaw table.

Refer to the *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3, or the online man page for more information about disk maintenance commands.

Refer to the *UNICOS Basic Administration Guide for CRAY J90 and CRAY EL Systems*, Cray Research publication number SG-2416 8.0.3.3, for information about disk drive administration.

Tape Drives

The CRAY J90 series tape systems all use a SCSI controller. The master input/output processor (IOP), usually IOS 0, provides a SCSI port for maintenance devices. The digital audio tape (DAT) drive is connected to this SCSI port. It is connected to the daughter board on the P2 connector located on the back of the VME backplane. Other tape drives are controlled by SI-3 controllers in the VME IOS chassis.

Tape Drive and Tape Controller Tests

You can use the following tape diagnostics to test tape drives:

- UNICOS online tape test `unitap`
- IOS based offline diagnostic `tp1test`

Tape Error Information

You can locate disk error information by using any of the following methods:

- Use the UNICOS `olhpa` command (refer to page 27)
- Use the UNICOS `errpt` command (refer to page 28)
- Examine the IOS `/adm/syslog` file

Networks

CRAY J90 series systems use two types of network controllers: Ethernet and fiber distributed data interface (FDDI). You can use the following diagnostics and utilities to test Ethernet and FDDI network controllers:

- IOS based offline diagnostic `nettest`
- IOS based utility `enstat` (Ethernet controller status)

HIPPI Channel

Use the `vht` command to troubleshoot the HIPPI channel. The following example shows how to use the `vht` command.

```
sn9099 (root)#
sn9099 (root)# vht -i /dev/hippi0/ioo -o /dev/hippi0/o00 -c1000 -D -P -1
Using I-field 0000000
random pattern selection
HIPPI test path 0 completed pass 800

vht 0 passes 1-1000 no fata errors
```

UNICOS Commands

You can use the following UNICOS commands when you are troubleshooting peripherals:

- `unitap` - online tape diagnostic
- `olhpa` - online hardware performance analyzer

CPUs and Memory

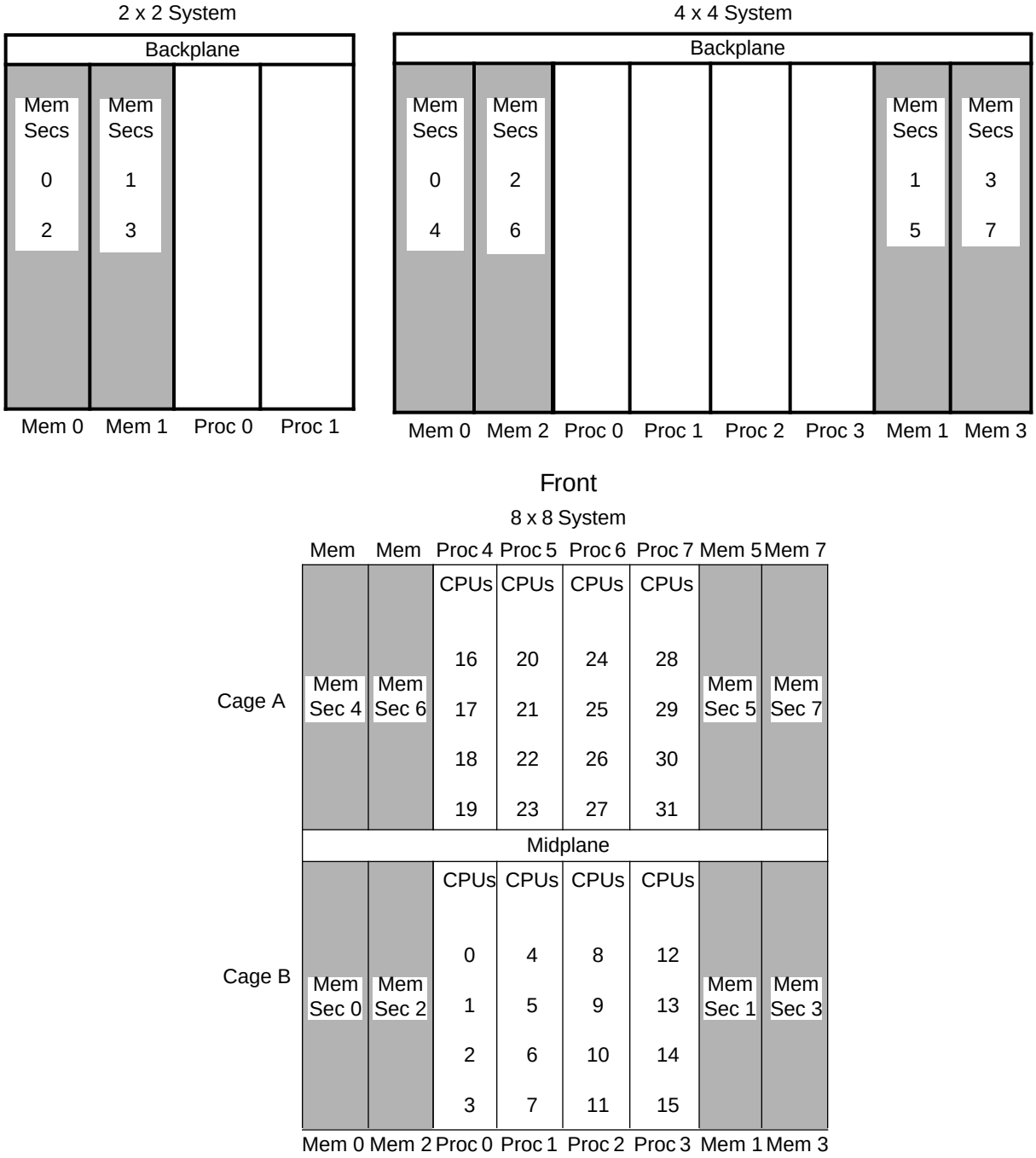
This section describes the troubleshooting techniques that you can use to troubleshoot CPU or memory failures. Refer to Figure 9 for an illustration of the 2 x 2, 4 x 4, and 8 x 8 processor and memory configurations.

If UNICOS is operating, you can use mainframe online diagnostics to test the CPUs and memory while UNICOS is active. Typically, you will troubleshoot mainframe problems in an offline environment. Begin mainframe memory or CPU troubleshooting by using the ACT menu system. If you are familiar with Cray Research offline diagnostics, you can use the offline diagnostic programs to isolate failures.

You should have also run the system boundary scan test `jbs` from the `snxxxx-ios0>` prompt to rule out any system interconnect defects. The boundary scan test does not test the functions of the internal logic on the various ASICs in the system; it tests only the input and output interconnections.

You should run mainframe offline diagnostics when you suspect that a failure is related to a defective CPU or a specific section of mainframe memory.

Figure 9. Processor and Memory Module Slot Locations



NOTES:

Proc 1 slot may be vacant in 2 x 2 configurations.
Proc 1, Proc 2, and Proc 3 slots may be vacant in 4 x 4 configurations.
Proc 4, Proc 5, Proc 6, and Proc 7 slots may be vacant in an 8 x 8 configuration.
All memory module slots will always be filled.

Memory Address Bit Assignments

Each chassis has different memory bit address assignments; Table 8 provides the absolute address and logical address for each chip, bank, subsection, and section for an 8 x 8 midplane, 4 x 4 backplane, and 2 x 2 backplane.

Table 8. Memory Bit Address Assignments

Area	8 x 8 Midplane		4 x 4 Backplane		2 x 2 Backplane	
	Absolute	Logical	Absolute	Logical	Absolute	Logical
Chip	31 - 10	31- 10	31 - 10	30 - 9	31 - 10	29 - 8
Bank [c]	9 8 7 6	9 8 7 6	9 8 7 6	8 7 6 5	9 8 7 6	7 6 5 4
Subsection	5 4 3	5 4 3	5 4 3	2 4 3 [a]	5 4 3	2 1 3 [b]
Section	2 1 0	2 1 0	2 1 0	2 1 0	2 1 0	2 1 0

[a] 1 bit of subsection address is hard wired.

[b] 2 bits of subsection address are hard wired.

[c] Half-populated modules use only the lower 3 bits for the bank address.

Using the ACT Menu System

You can test specific areas of the mainframe by using the ACT menu system. Enter `act_menu` at the `sn`*`xxxx`*`-ios0>` prompt to invoke the ACT menu system. Two options under the basic test menu (refer to Figure 10) can be used to deaddump central memory or to test a single processor board or CPU.

Figure 10. ACT Basic Test Menu

```
Automated Confidence (BASIC) Test Menu.

  NOTE: These tests must complete successfully before
        you run the next level of tests. Not following
        this sequence could lead to an erroneous FRU
        being detected as the fault.

  1) Run All Basic Test.
  2) Test IOS, IOBB and Peripherals.
  3) Run System Boundary Scan.
  4) Deaddump Central Memory Test.
  5) Single Processor Board, Single CPU Test.
  6) Test IOS Y1 Channel Control IOBB <-> CM.

  n) Next level tests (intermediate) menu.
  h) Help.
  q) Quit.
Enter selection:
```

Mainframe Online Diagnostics

Online diagnostics enable you to ensure the reliability of Cray Research computer systems without having to power down the system for maintenance. The online diagnostics are a set of programs that detect, isolate, and report hardware faults; that gather and analyze data; and that can concurrently bring down a failing CPU without affecting the rest of the system.

Refer to the *CRAY Y-MP, CRAY X-MP EA, and CRAY X-MP Computer Systems UNICOS Online Diagnostic Maintenance Manual*, Cray Research publication number SPM-1012 8.0, for descriptions of the UNICOS online diagnostic commands.

CPU and Memory Diagnostics

Table 9 lists the CPU and memory online diagnostic confidence tests. The online diagnostic programs reside in the `/ce/bin` directory under UNICOS.

Table 9. UNICOS Online Diagnostic CPU and Memory Tests

Test	Description
Confidence Tests	
olcfpt	Comprehensive floating-point test
olcm	Central memory test (single-CPU mode only)
olcrit	Comprehensive random instruction test
olcsvc	Comprehensive scalar and vector comparison test
olibuf	Instruction buffer test
olsbt	Semaphore, shared B and T register test
Maintenance Tests	
olyfpt	Floating-point add, multiply, and reciprocal approximation functional unit test
olymem1	Memory paths via scalar and vector memory references test
olypave	Memory addressing, conflicts, and port selection test
olysas	A and S register instructions test that uses simulated results
olyscl	Scalar logical instruction 42 through 51 test
olyscs	Scalar register shift test
olysr2	Register and scalar instruction (3-parcel) conflicts test
olysr3	Register and scalar/vector (3-parcel) instruction test
olyvpt	Random vector register and path test

Online Tape Test

The UNICOS online magnetic tape diagnostic environment, `unitap`, is an interactive, menu-driven, online tape test that provides the following testing options:

- Quick confidence tests. These are short, predefined test sequences that test the basic functions of the tape drive. These tests take approximately 30 seconds to complete.
- All confidence tests. The `unitap` environment executes all of the tape tests. These tests take approximately 3 minutes to complete.
- Two- through eight-path conflict test. This test runs a selection of tape tests in parallel to exercise from two to eight tape paths.
- Canned tests. These tests run user-selected tests.
- Test loop. A user-defined test loop can be written with the `unitap` program tool.

Refer to the *CRAY Y-MP, CRAY X-MP EA, and CRAY X-MP Computer Systems UNICOS Online Diagnostic Maintenance Manual*, Cray Research publication number SPM-1012 8.0, for more information about `unitap`.

Mainframe Offline Diagnostics

The CRAY J90 series mainframe offline diagnostics are modified versions of CRAY Y-MP style offline diagnostics. In order to successfully use the mainframe offline diagnostics, you must have a thorough knowledge of CRAY Y-MP style offline diagnostics and you must also have access to the diagnostic listings.

The `offline` utility loads and configures the diagnostic tests that are stored on the system disk. The offline utility should be used only by escalated support personnel. You should use the ACT menu system and the boundary scan test, `jbs`, before you attempt to run and interpret failure information from mainframe offline diagnostics.

Mainframe Offline Diagnostic Listings

Mainframe offline diagnostic listings and all of the CRAY J90 series hardware maintenance documents are provided by accessing the Service Publications and Training Web site at `http://servinfo.cray.com/j90`. If you have additional questions about the CRAY J90 series online hardware information or the diagnostic listings, contact the Service Publications and Training department at the following address, or send e-mail to `online@cray.com`.

Attn: Service Publications and Training Online Team
890 Industrial Boulevard
P.O. Box 4000
Chippewa Falls, WI 54729-0078
USA

Utilities Quick Reference

Table 10 provides a quick reference of the mainframe offline utilities.

Table 10. Mainframe Offline Utilities

Name	Test Description
jclr	Clear the interrupts, memory, cache, and registers utility
jdump	Logical master CPU's B, T, vector, SB, ST, and semaphore register to memory dump utility
jexd	Exchange package dump utility
jfind	Memory location find utility
jmnum	Memory address pattern utility

Diagnostics Quick Reference

Table 11 provides a quick reference of the CRAY J90 series mainframe diagnostic descriptions.

Table 11. Mainframe Offline Diagnostic Descriptions

Name	Test Description
jaab	A register functions and the A integer adder
jaat	Basic address-adder test
jaht	A*h* addressing test. Checks A register-to-memory addressing
jamb	Address multiply basic test
jars	Address and scalar registers add and multiply test
jave	Vector register test
jbrt	Block transfer register test
jbsr	Basic shared and semaphore registers test
jbtas	B-to-T register transfer or A-to-S register transfer test
jcht	Y1 channel test
jcrid	Multiple-CPU, random-instruction test
jejt	Exchange jump test
jexch	Exchange package functions utility
jfpt	Floating-point units test
jhpi	HIPPI channel basic test
jibt	Instruction buffer test
jiot	Comprehensive multi-CPU based I/O test
jjpt	Jump test
jpath	DS ASIC data transfer test
jpave	Memory addressing, conflicts, and port select test
jpmt	Performance monitor test
jsab	Scalar register basic test
jsas	Scalar adder test
jscl	Scalar logical basic test
jscs	Scalar shift test
jsct	Scalar cache test
jsdt	Error correction and detection test
jsem	Shared and semaphore registers test
jsfa	Floating-point add test
jsfm	Floating-point multiply functional unit test
jsfr	Floating-point reciprocal functional unit test
jsr2	Register and scalar instruction conflicts test (3-parcel) using random operands
jsr3	Scalar and vector registers (3-parcel) conflict test
jvbt	Vector register basic test
jvpt	Vector path test
jvsg	Scatter/gather instruction test

Troubleshooting with Offline Diagnostics

Table 12 lists the mainframe offline diagnostics by function to help you determine which diagnostics to use to check out areas of the mainframe.

Table 12. Troubleshooting with Offline Diagnostics

Primary Functional Sets	Diagnostics
A, S, and V Registers	jaab
	jave
	jsab
	jvbt
	jvpt
B and T Registers	jbrt
	jbtas
Confidence	jcrid
	jiot
	jpath
Control	jsr2
	jsr3
	jbsr
	jejt
	jibt
	jjpt
Semaphore and Shared Registers	jsem
Functional Units	jamb
	jars
	jfpt
	jsas
	jscl
	jscs
	jsfm
	jsfr
	jvpt
Memory	jaht
	jpave
	jvsg
I/O	jiot
	jcht

Using the Offline Command

Escalated support personnel can use the `offline` command to load, configure, and run mainframe offline diagnostics in all CPUs simultaneously; to execute a diagnostic in any one selected CPU; or to run a diagnostic in all CPUs and designate any CPU as master.

The `offline` command loads and automatically configures mainframe diagnostics. The `ds` (deadstart) command runs the diagnostic in **all** CPUs. The default master CPU is CPU 0. The `offline` command uses the `/config` file to determine the number of CPUs in the system.

The `offline` command is invoked from the `snxxxx-ios0>` prompt, and it obtains configuration information from the IOS `/config` file. The `offline` command options enable you to specify a monitor, a particular set of CPUs, memory size, etc. Use the IOS `man` command to review all the available options for the `offline` command.

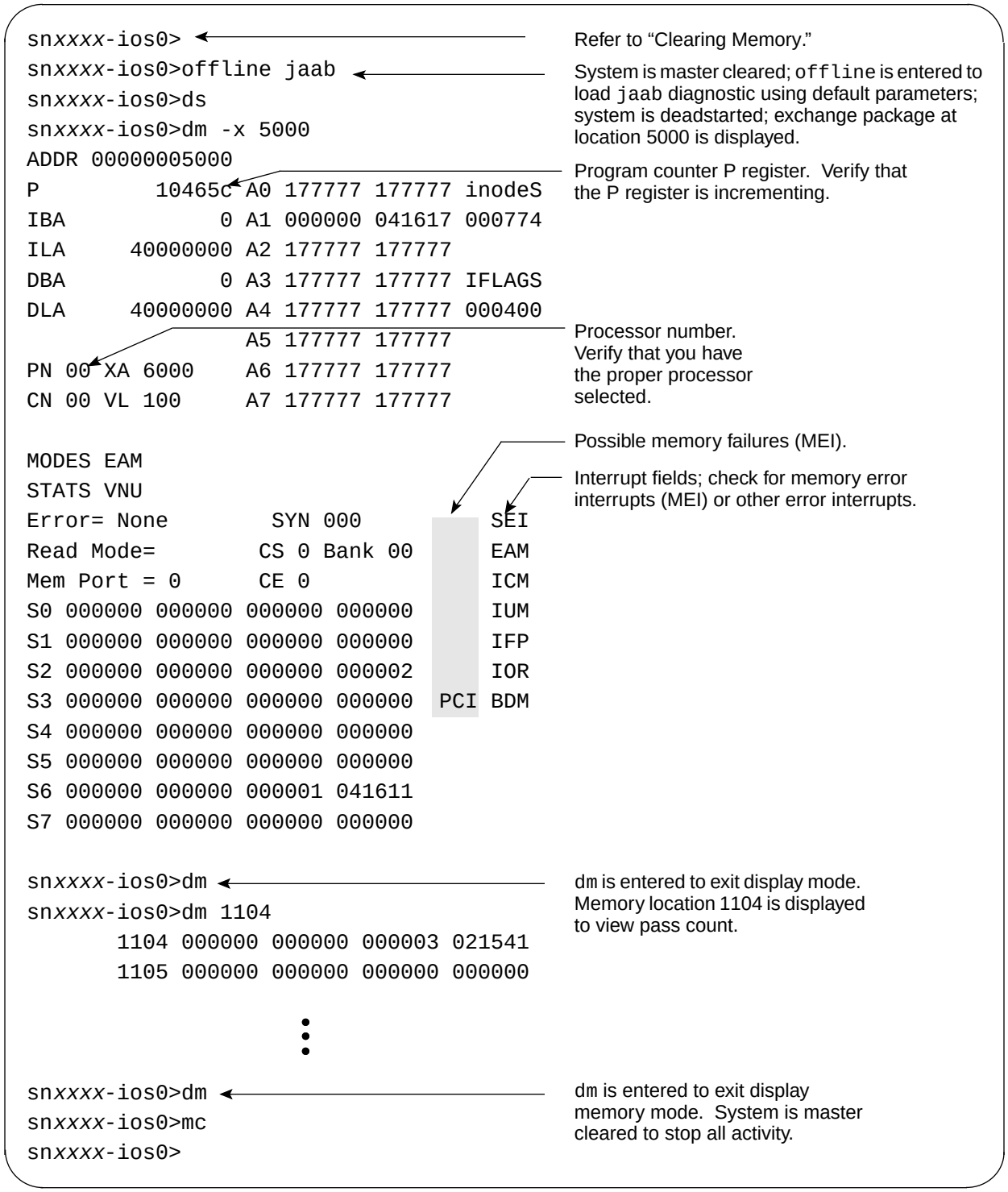
In order to use the `offline` command, you must also be familiar with the `dm` (display memory), `ds` (deadstart), and `mc` (master clear) commands. You must also have access to the offline diagnostic listings.

NOTE: The display of central memory is continuous until a `dm` command (without options) is executed, or until a `q` is entered to quit the display. The memory display must be stopped *before* you enter any command other than `dm`.

Figure 11 shows a sample output of these commands. The highlighted areas show status information that is important when you are troubleshooting.

NOTE: Use the `adrm -o` command to set the memory display to octal before using the `dm` command.

Figure 11. Offline Diagnostics Output



Offline Command Examples

The following subsections provide examples of how you can use the `offline` command. The following list shows standard memory locations:

Location	Description
1100	Differences, CPU 0 or master
1101	Actual data, CPU 0 or master
1102	Expected data, CPU 0 or master
1103	Fail count, CPU 0 or master
1104	Pass count, CPU 0 or master
1105	Error return jump address, CPU 0 or master
1106 through 1107	Open
1110 through 1117	CPU 1
1120 through 1127	CPU 2
1270 through 1277	CPU 15
5000 through 5017	CPU 0 or master exchange package
5020 through 5360	CPUs 1 – 15 current exchange package
1700 through 1777	Memory error table (default)
2000	Start of test code

Clearing Memory

You must clear memory with the `jclr` utility before you run any tests on mainframe memory. The following example shows the `jclr` command loaded and deadstarted with the `offline` and `ds` commands. The `dm` command verifies that the `jclr` command has completed one pass. The `jpave` memory diagnostic is then loaded and deadstarted.

```
snxxxx-ios0> mc
snxxxx-ios0> offline jclr
snxxxx-ios0> ds
snxxxx-ios0> dm 1100
snxxxx-ios0> dm
snxxxx-ios0> mc
snxxxx-ios0> offline jpave
snxxxx-ios0> ds
```

Changing the Master CPU

The `offline` command, when used without any parameters (default), loads all CPUs. The `ds` (deadstart) command executes the diagnostic in all CPUs. The default master CPU is 0. In the example below, the `jaab` diagnostic will execute in all CPUs. The `stat` command will show that all CPU P registers are incrementing.

NOTE: You must enter the processor module slot number and pass count with the `stat` command.

```
snxxxx-ios0> (refer to the previous subsection "Clearing Memory")
snxxxx-ios0> mc
snxxxx-ios0> offline jaab
snxxxx-ios0> ds
snxxxx-ios0> stat 0 20
```

In the following example, the master CPU is changed from CPU 0 to CPU 1. The `ds` command is then executed with an argument of 1 (to start CPU 1 first).

```
snxxxx-ios0> (refer to "Clearing Memory")
snxxxx-ios0> mc
snxxxx-ios0> offline jaab
snxxxx-ios0> ds 1
```

Viewing CPU Exchange Packages

To view an exchange package, use the `dm` command to examine the following memory locations as shown below. Refer to the *CRAY J90 Series System Programmer Reference Manual*, Cray Research publication number CSM-0301-0A0, for more information about the exchange package.

NOTE: Be sure to use the `adrm-o` command to set the addressing mode to octal before using `dm`. Set the addressing mode back to decimal, which is the default, when finished (`adrm d`).

CPU 1 and CPU 0 change places in the exchange package area of memory. The master CPU always uses memory location 5000. The following example displays the exchange at location 5000 for CPU 1. The PN (processor number) field should equal 1 because of the ds 1 command that was issued earlier.

```
snxxxx-ios0> dm -x 5000
```

To display the exchange package at location 5020 (CPU 0) on the right side of the display, enter the following command. The PN field should equal 0.

```
snxxxx-ios0> dm -rx 5020
snxxxx-ios0> dm
```

CPU 3 is the master CPU in the following example. The pass count at location 1104 is displayed, as well as the exchange package at location 5000. The master CPU's values are displayed in both the exchange package at 5000 and in the pass count at location 1104. The PN field at location 5000 should equal 3.

```
snxxxx-ios0> (refer to "Clearing Memory")
snxxxx-ios0> mc
snxxxx-ios0> offline jaab
snxxxx-ios0> ds 3
snxxxx-ios0> dm 1104
snxxxx-ios0> dm -rx 5000
```

Refer to the following screen snap for an example of the exchange package.

```
ADDR 00000005000
 P          14255c A0 000000 000001 inodeS
 IBA             0 A1 000000 000011 000777
 ILA    4000000000 A2 000000 000034
 DBA             0 A3 000115 000115 IFLAGS
 DLA    4000000000 A4 000000 000044 000000
                   A5 000000 044710
 PN 00  XA 6000    A6 000040 000000
 CN 00  VL 100     A7 000000 000000

 MODES EAM                          MM
 STATS VNU                          IMM
 Error= None         SYN 000        SEI
 Read Mode=        CS 0 Bank 00     EAM
 Mem Port = 0      CE 1             ICM
 S0 000000 000000 000000 000001     IUM
 S1 000000 000000 000000 000001     IFP
 S2 000000 000000 000000 000001     IOR
 S3 000000 000000 000000 000044     BDM
 S4 066554 104577 020433 000765
 S5 052011 055752 016162 036424
 S6 077715 154132 016441 131717
 S7 105506 004423 111251 117454
```

Executing in Selected CPUs - Including CPU 0

To execute a diagnostic in a set of selected CPUs, specify the CPUs by using the `-c` option and an octal bit mask value. In the following example, CPUs 1 and 2 have been masked out (of an 8-CPU system) with a bit mask value of 371. CPU 0 is the default master CPU. The `stat` command will show that CPUs 1 and 2 are not running.

CPU Number	-	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Octal Bit Mask	-	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1

sn*xxxx*-ios0> (refer to “Clearing Memory”)
sn*xxxx*-ios0> **mc**
sn*xxxx*-ios0> **offline -c 371 jaab**
sn*xxxx*-ios0> **ds**
sn*xxxx*-ios0> **stat 0 20**

Executing in Selected CPUs - Not Including CPU 0

To execute a diagnostic in a set of selected CPUs that does not include *physical* CPU 0, you must specify a *logical* CPU 0, or master. To do this you must use a bit mask that includes CPU 0 (because it is really a logical bit mask), and then deadstart the alternative master CPU.

NOTE: Because you are assigning the master CPU the logical CPU 0 designation, the physical number of the master CPU is taken out of the bit mask.

In the following example, the bit mask value of 341 in combination with the `ds 4` command will test CPUs 4 through 7, with CPU 4 as the logical master CPU 0. Keep in mind that physical CPU 4 is not part of the selection because it was assigned the master CPU (CPU 0) designation. The `stat` command will show that only CPUs 4 through 7 (in processor module slot 1) are running.

CPU Number	-	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Octal Bit Mask	-	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1

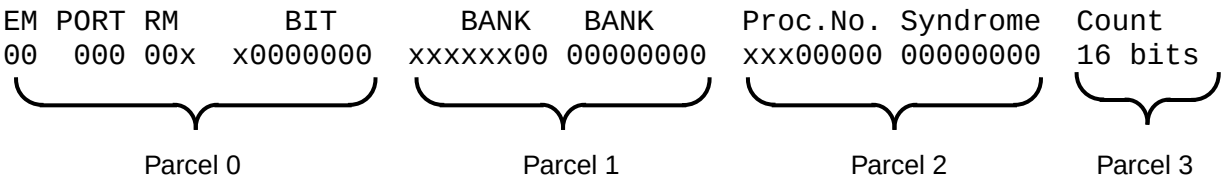
```
snxxxx-ios0> (refer to “Clearing Memory”)
snxxxx-ios0> mc
snxxxx-ios0> offline -c 341 jaab
snxxxx-ios0> ds 4
snxxxx-ios0> stat 1 20
```

Displaying Memory Error Information

Memory error information is displayed at location 1700. Use the `dm` command to view this location if memory errors occur. To determine which memory section (module) is failing, decode the last 3 bits of parcel b. Bits 0 and 1 indicate the section. The field replaceable unit (FRU) in the following example is section 3, (memory module 1).

```
snxxxx-ios0> dm 1700

Parcel   01  23
1700   134508   137073   000531   050076
Parcel 1 = 137073 (1  011  111  000  111  011)
```



`Count`	-	Total count	Bits	0 through 15
`Syndrome`	-	Syndrome number	Bits	16 through 23
`Proc.No.`	-	Processor number	Bits	24 through 28
	-	Unused	Bits	29 through 31
`BANK`	-	Bank	Bits	32 through 41
	-	Unused	Bits	42 through 47
`BIT`	-	Bit number	Bits	48 through 54
	-	Unused	Bits	55 through 56
`RM`	-	Read mode	Bits	57 through 58
`PORT`	-	Port	Bits	59 through 61
`E`	-	Error mode	Bits	62 through 63

`BIT`	-	Bit number	Bits	48 through 54
	-	Unused	Bits	55 through 56
`RM`	-	Read mode	Bits	57 through 58
`PORT`	-	Port	Bits	59 through 61
`E`	-	Error mode	Bits	62 through 63

Using IOS Based Tests and Utilities

Table 13 lists IOS based tests and utilities that you can use to test CPUs and memory. Refer to the online manual (man) pages for more information about each test or utility.

NOTE: The accuracy of the ACT FRU list depends on a specific sequence of testing. Always use the ACT menu system and complete all of the tests on each menu level before you proceed to the next menu level. Running any of the ACT executable programs out of sequence may result in an erroneous failing FRU list.

Table 13. IOS Based Mainframe Tests and Utilities

Test/Utility	Description
act_menu	ACT menu system should be the first choice when troubleshooting the mainframe
cc1test	Uses data channel I/O to check the related interfaces between the IOP, IOBB, and CM
cc2test	Checks the related interfaces that control CM-to-IOBB-to-CM data transfers
dm	Displays central memory
ds	Loads and deadstarts a mainframe offline diagnostic
jbs	Boundary scan test
offline	Loads and configures a mainframe offline diagnostic
stat	Displays CPU status and activity

Using the stat Command

The `stat` command displays CPU program states for each CPU that is configured at boot time in the given processor board slot. Refer to the *CRAY IOS-V Commands Reference Manual*, Cray Research publication number SR-2170 8.0.3, or the online man page for more information about the `stat` command.

You can enter the `stat` command while offline diagnostics are running to view the activity of each CPU. In the example shown below, the number 0 specifies the processor board slot number. The number 10 specifies the number of times the `stat` function is executed before the system returns to the `snxxxx-ios0>` prompt.

```
sn9004-ios0>stat 0 10

Slot 0

Cpu                 0            1            2            3
Preg                00000000325c 00000010226c 00000010210c 00000010226c
CIP                 00000037300  00000176104  00000176104  00000176104
Modes                      0011         0770         0770         0770
Flags                      0000         0000         0000         0000
MM                            1            0            0            0
Cln                          00           00           00           00
VL                          100          100          100          100
XA                         0000         0020         0040         0060

sn9004-ios0>
```

System Configuration Utility jconfig

NOTE: The `jconfig` utility should be used only by escalated support personnel. Corrupted system configuration files make the system inoperable.

The `jconfig` utility enables you to build and edit CRAY J90 series hardware configuration files. This utility builds one configuration file for each processor module and one configuration file for each memory module in the system. These files are named `/sys/pm`*[0-7]*`.cfg` for processor modules and `/sys/mem`*[0-7]*`.cfg` for memory modules. These files are then used during master clear (`mc` command) and deadstart (`ds` command) sequences.

The `.cfg` files contain all configuration data needed by the configuration registers contained in every ASIC test access port (TAP) controller. The `mc` and the `ds` commands read the `.cfg` files and shift the configuration data into the TAP controllers.

NOTE: These `.cfg` files are not used by the boundary scan test `jbs`.

jconfig Sequence

When `jconfig` is run in the default mode (no command line options), `jconfig` performs the following sequence:

1. Resets the CRAY J90 series system, so that it can perform maintenance functions.
2. Reads the system configuration to determine which slots have modules in them.
3. Attempts to read an existing `.cfg` file to extract nonhardware-readable information (backplane type and memory module type codes).
4. If either Step 2 or Step 3 fails, you can enter slot, backplane, and memory module type codes by using an information screen that you can edit with a `vi`-type editor.
5. Displays the main menu.

From this point, you can edit the configuration data, view the current hardware configuration, update the system `.cfg` files, and dump the `.cfg` files to ASCII files.

Command Line Syntax

```
jconfig   [-iobb <devname>] [-edit] [-nocr] [-help]
          [-ecc <on | off>] [-cache <on | off>]
          [-mm <on | off> [-degrade <even | odd | none>
          [-bpmt <backplane memory_types>
          [-hwconfig <cpu_bitmap backplane memory_types>]
```

`-iobb <devname>`

This option is used to test or run `jconfig` in a Systems Test and Checkout test vehicle environment. This option enables you to specify a specific IOBB device other than the default `/dev/iobb`.

`-edit`

Enables `jconfig` to run without performing a hardware read or check. This option can be used to edit the `.cfg` file data or when the hardware is defective.

`-nocr`

Used only with `-bpmt` or `-hwconfig`. Allows `jconfig` to be run from the command line with no user interaction. Normally, `jconfig` enters a menu interface, and there are times when `jconfig` prompts you to confirm choices by pressing the Return key. The `-nocr` option bypasses the menus and does not wait for a Return. This option is useful for running `jconfig` from command scripts. Refer to the `-bpmt` and `-hwconfig` command options.

`-help`

Prints all command line options.

The following options are inline options. If these options are specified on the command line, `jconfig` is invoked with no hardware check or editing capabilities. `jconfig` reads all `.cfg` files, alters the desired parameter system wide, and updates the .`cfg` files.

`-ecc <on | off>`

Set SECDED error-correction mode ON or OFF, system wide.

`-cache <on | off>`

Enable or disable scalar cache, system wide.

`-mm <on | off>`

Enable or disable Maintenance Mode instructions, system wide.

`-degrade <even | odd | none>`

Degrades system memory so that only EVEN sections will be accessed (even), only ODD sections will be accessed (odd), or ALL sections will be accessed (none).

NOTE: Using the `-degrade <even | odd | none>` option will not allow UNICOS operation.

The following options enable you to run `jconfig` with less user interaction.

`-bpmt <backplane memory_type_codes>\`

Specifies the backplane type and memory module type codes. Typically used when you run `jconfig` from a script. Backplane types are 1 x 1, 2 x 2, 4 x 4, and 8 x 8. Memory module type codes are single-digit hexadecimal codes that are read from a sticker on each memory module faceplate. Valid codes are 0, 1, 2, 3, 4, 5, 8, 9, a, b, c, and d. The codes entered are ordered beginning with memory module 0.

Example: `jconfig -bpmt 4 X 4 0 0 1 0`

This example specifies a 4 x 4 backplane and type code 0 for memory module 0, 1, and 3, and a type code of 1 for memory module 2.

`-hwconfig <cpu_bitmap memory_bitmap backplane memory_type_codes>`

This option enables you to enter ALL hardware parameters on the command line. `jconfig` does not perform a hardware check. Used together with `-nocr`, `jconfig` runs without user interaction.

`cpu_bitmap` is a hexadecimal bit map of all CPUs in the system. Each processor module has up to 4 CPUs on it. Bit 0 corresponds to CPU 0, bit 1 corresponds to CPU 1, and bit 31 corresponds to CPU 31. Therefore, for a 4 X 4 system with all CPUs installed, the CPU bitmap would be `ffff`.

`memory_bitmap` is a hexadecimal bit map of all the memory modules in the system. Bit 0 corresponds to memory module 0, bit 1 corresponds to memory module 1, and so on. For a 4 x 4 system with all memory modules installed, the memory bitmap would be `f`.

`backplane` is the backplane type. Valid backplane types are 1 x 1, 2 x 2, 4 x 4, and 8 x 8.

`memory_type_codes` are single-digit hexadecimal codes that are read from a sticker on each memory module faceplate. Valid codes are 0, 1, 2, 3, 4, 5, 8, 9, a, b, c, and d. The codes entered are ordered beginning with memory module 0.

Example: `jconfig -hwconfig ffff f 4x4 0 0 1 0`

This example specifies 4 processor modules with 4 CPUs on each module, 4 memory modules, a 4 x 4 backplane, memory module type codes 0 for memory modules 0, 1, and 3, and memory module type code 1 for memory module 2.

Configuration Information

The `jconfig` utility requires hardware configuration information such as which CPU slots have processor modules in them, which memory slots have memory modules in them, the number of CPUs per processor module, the backplane type, and the memory module type codes. Software can use the IOSNET (refer to page 9) to access the CPU and memory slot information and the number of CPUs per processor module. The backplane type and memory module type codes cannot be accessed.

The `jconfig` utility attempts to read the slot configuration if `jconfig` is invoked without the `-hwconfig` or `-bpmt` command line options. Then `jconfig` searches for a `.cfg` file. If one is found, `jconfig` reads the backplane type and memory module type codes for all memory modules. If either one of these operations fails, `jconfig` informs you and displays an information screen that you may edit to reflect the actual hardware configuration as shown in the following example:

```
  CP Boards/CPUs:    1 HEX Digit Per Board, 1 Bit Per CPU On That CP.
                     Example:   0000ffff  (CP 0-3, all cpus present/CP)
                                00000001  (CP 0, cpu0 only present)

  MEMORY Boards:     1 Digit Per Board.  1 == Mem. Board Present, 0 == Not Present

  Memory Type:       1 Digit Per Memory Board.  Get Type Code From Memory Board
                     Sticker.  Valid Type Codes Are 0-5 and 8-d (hexadecimal).

  Backplane Type:    8==8x8, 4==4x4, 2==2x2, 1==1x1
=============================================================================

  CP Boards/CPUs:    00000000   (Leftmost Digit is CP  Board 7, Rightmost is CP 0)
  MEMORY Boards:     00000000   (Leftmost Digit is MEM Board 7, Rightmost is MEM 0)
  Memory Type:       00000000   (Leftmost Digit is MEM Board 7, Rightmost is MEM 0)
  Backplane Type:    00000000   (8==8x8, 4==4x4, 2==2x2, 1==1x1)


    <h,l,k,j,CR,p> Left,Right,Up,Down,Next Line,Page   <z> Save   <ESC> Discard
                                                                  [Page 1 of 1]
```

You may use the editor commands to enter information (refer to“Editing jconfig Parameters” on page 77) and enter `z` to save the configuration. At this point, `jconfig` prompts you for verification of the information entered. If no errors are encountered during hardware configuration sensing, `jconfig` prompts you for verification of the hardware configuration, and then displays the main menu.

Main Menu

The main menu is displayed if `jconfig` is invoked without the `-nocr` option and `jconfig` has determined the hardware configuration. These menus enable you to select the type and scope of ASIC configuration register fields that can be edited.

```
==============================MAIN_MENU==============================


          <1> Edit Diagnostic Parameters
          <2> Edit ALL Parameters
          <3> View System Configuration
          <4> Update Config File(s)
          <5> Exit

          Enter # Of Choice:
```

Editing jconfig Parameters

NOTE: Editing the system configuration parameters may make the system inoperable. System configuration parameters should be edited only by escalated support personnel.

`jconfig` enables you to edit all fields of all ASIC configuration registers. These fields are grouped into two types: `Diagnostic` (fields that can be used to diagnose problems on a CRAY J90 series system or alter the way the system runs) and `All` (any and all configuration fields). Each field can also be controlled by scope. You can alter certain parameters for the entire system, a specific module, a specific ASIC type on a module, or a specific ASIC type on the entire system.

NOTE: Typically, the system-wide diagnostic settings are the most appropriate to use.

The edit screens display each configuration register field, an explanation of what the field is for, and how many bits it occupies. All fields are expressed as 8-digit hexadecimal numbers. A footer at the bottom of the screen has editing instructions, the current field type and scope, and the current edit page number.

The cursor may be moved with the same cursor keys as the `vi` editor. Entering `z` saves the edit(s), after which the main menu is displayed. To discard edits, press the escape key (Esc).

```
Disable Error Correction (SECDED) (1==Disable)1  Bit:  00000000
Set Maintenance Mode System Wide   (1==Set)1  Bit:  00000000
Disable Scalar Cache System Wide   (1==Disable)1  Bit:  00000000
Memory Degraded (Odd or Even Sects Only, 1==Degraded)1  Bit:  00000000
Even Section Only or Odd Sections Only (1==even)1  Bit:  00000000
Physical CPU # of Logical CPU 0 (0-1f)5 Bits:  00000000
Disable 005 During 034, System-wide PC's (1==disable)1  Bit:  00000001
Disable 024 During 036, System-Wide PC's (1--disable)1  Bit:  00000001
Wait On Data During 073/076, System-Wide PC's1  Bit:  00000000
Allow 1 Instr. To VU At A Time, System-Wide PC's 1  Bit:  00000000
Allow Only 1 Port Active At a Time, System-Wide PC's1  Bit:  00000000
Disable Instr. Chaining, System-Wide VU's (1==disable)1  Bit:  00000000
Disable Instr. Tailgating, System-Wide VU's (1==disable)1  Bit:  00000000
Disable 1 CP Bypass, System-Wide VU's (1==disable)1  Bit:  00000000
Disable 2 CP Bypass, System-Wide VU's (1==disable)1  Bit:  00000000
Logical CPU board Number, Physical CPU Board 0 (0-7)3 Bits:  00000000
Logical CPU board Number, Physical CPU Board 1 (0-7)3 Bits:  00000001
Logical CPU board Number, Physical CPU Board 2 (0-7)3 Bits:  00000002
Logical CPU board Number, Physical CPU Board 3 (0-7)3 Bits:  00000003
Logical CPU board Number, Physical CPU Board 4 (0-7)3 Bits:  No Board

<h,l,k,j,CR,p> Left, Right, Up, Down, Next Line, Page<z> Save  <ESC> Discard
PARAMETER GROUP: Diagnostic System Level     [Page 1 of 2]
```

Saving and Updating Configuration Files

The `jconfig` utility enables you to save the current configuration files (`.cfg` files) and ASCII files using the `Update Config File(s)` option from the main menu.

NOTE: The `.cfg` files are saved automatically if the `-nocr` command line option is used.

Return Values

The `jconfig` program returns a zero to the calling environment if it runs without error; it returns a nonzero if an error occurs.

Error Messages

Error messages are generated if `jconfig` has trouble sensing the hardware configuration, or if errors are encountered during the file open function, file read function, or file write functions. All error messages start with `jconfig:`. Fatal errors result in messages that start with `jconfig: Aborting`.

Files

Table 14 lists the files that `jconfig` uses.

Table 14. jconfig Files

File	Description
/sys/pm[0-7].cfg	Module-level processor module configuration files that contain ASIC JTAG configuration register fields for that module
/sys/mem[0-7].cfg	Module-level memory module configuration files that contain ASIC JTAG configuration register fields for that module
/sys/pm[0-7].cfg.txt	ASCII version of `pm[0-7].cfg`
/sys/mem[0-7].cfg.txt	ASCII version of `mem[0-7].cfg`