System Troubleshooting

HMM-307-0
CRAY J90se Series Systems
Last Modified: March 1997

Figures

Tables

Record of Revision

The latest revision of this manual is available on the Service Publications and Training Web server at `http://servinfo.cray.com`.

March 1997

Original printing.

How to Use this Document

This document describes basic hardware and software troubleshooting techniques for service personnel who have had CRAY J90se series training.

To use this document effectively, the troubleshooter must be familiar with the topics covered in the “Maintenance Environment” section and then proceed to the “System-level Troubleshooting” section to determine which part of the system is failing.

Refer to the *SIO Troubleshooting Guide*, for information about troubleshooting the scalable I/O architecture. The “Mainframe Troubleshooting” section provides information about troubleshooting the CRAY J90se series mainframe.

This document is divided into three sections:

- Maintenance Environment - This section covers background information and provides a description of the troubleshooting tools available on the system workstation (SWS).

- System-level Troubleshooting - This section includes a system-level troubleshooting flowchart, on page 13, that you can follow to determine which part of the system is failing (MPN-1, processor module, memory module, peripheral, controller, etc.) Continue to the appropriate section of this document or the *SIO Troubleshooting Guide* to check the status of the GigaRing channel.

- Mainframe Troubleshooting - This section describes the tools that are available for troubleshooting the mainframe using automated confidence testing, offline diagnostics, and boundary scan. The system configuration tool `jconfig` is also covered.

Maintenance Environment

This section introduces all of the maintenance tools that are available on the SWS and is organized into the following sections:

- System Workstation
- Diagnostics
- Error Information

System Workstation

The system workstation (SWS) serves as both a system administration platform and as a hardware maintenance platform. All the CRAY J90se series offline diagnostics reside on the SWS along with the diagnostic interface utilities.

NOTE: Log in as `craydiag` or `crayadm` to use the diagnostic utilities on the SWS.

Figure 1 shows the basic configuration of a CRAY J90se series system. Refer to the *PC-10 Installation and Cabling* document, Cray Research part number HMM-371-0, for detailed cabling information.

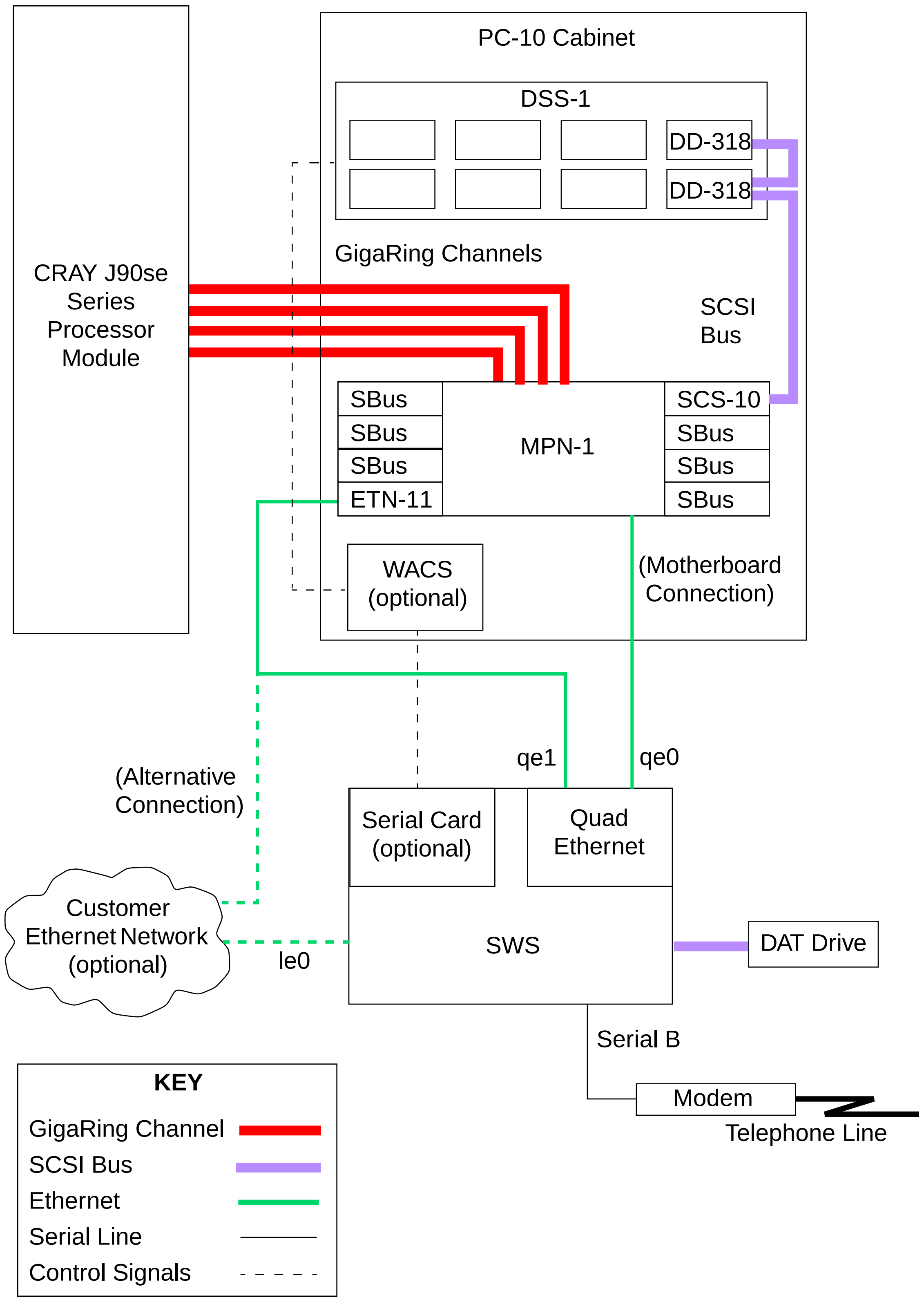
The SWS `qe0` Ethernet port is used as the maintenance channel. It connects to the maintenance host (typically the MPN motherboard Ethernet connection). The SWS uses this channel to assign the IP addresses to all the nodes on the GigaRing channel to upload their operating system software (VxWorks). The `qe1` or `le0` Ethernet connections can be used for customer Ethernet access to the SWS.

An optional serial port connection to the PC-10 warning and control system (WACS) is used for WACS operations. The serial B port is used for a remote support connection to the SWS.

The minimum disk drive configuration to support UNICOS is one SCSI SBus controller (SCS-10) that is connected to two DD-318 disk drives in a daisy chain configuration. Additional SBus contollers can be added to the MPN such as asynchronous transfer mode (ATM-10), Fiber Distributed Data Interface (FDI-10), and an Ethernet controller (ETN-11).

The four GigaRing connections from the MPN are connected to a processor module in the CRAY J90se series mainframe.

Figure 1. Block Diagram of a Basic CRAY J90se I/O Configuration



Diagnostics

This section provides an overview of the diagnostics and utilities that are available for troubleshooting CRAY J90se series systems. It is divided into the following subsections:

- UNICOS Online Diagnostics and Utilities
- Mainframe Offline Diagnostics
- Automated Confidence Testing (ACT)
- JMC

UNICOS Online Diagnostics and Utilities

UNICOS online diagnostics and utilities run concurrently with UNICOS. Refer to the *SWS-ION Administration and Operations Guide*, Cray Research publication number SG-2204, for more information about UNICOS online diagnostics.

Mainframe Offline Diagnostics

The primary tool for running offline mainframe diagnostics is the automated confidence testing (ACT) menu interface. The J90 maintenance console (JMC) can also be used to run offline mainframe diagnostics.

Automated Confidence Testing (ACT)

ACT is a suite of diagnostic tests that detect and isolate hardware failures in CRAY J90se series computer systems. ACT is a menu-driven interface that enables you to select and run specific diagnostics. ACT is also designed to provide field replaceable unit (FRU) isolation capabilities by encouraging the user to run tests in a specific order. Each segment can be used by on-site, remote support, or escalated support service personnel to troubleshoot the system.

To successfully troubleshoot your CRAY J90se series system, you must be able to interpret the failure information that ACT displays on the SWS and logs to error files.

Refer to page 24 for information about how to use ACT.

JMC

JMC is an offline mainframe diagnostic interface that is designed for use by Systems Test and Checkout (STCO) personnel. JMC is included in the standard SWS software package and can be used by field personnel as an advanced troubleshooting tool. JMC offers more flexibility than ACT; however, it requires a more detailed understanding of the hardware and does not provide FRU isolation capabilities.

Refer to page 39 for information about how to use JMC.

Error Information

The following sections describe the error reporting tools and the location error information:

- Important Files and Directories
- sysmon
- sserrpt
- errpt

Important Files and Directories

Table 1 lists some of the important files and directories that you must be familiar with before you troubleshoot the system.

Table 1. Important Files and Directories on the SWS

File or Directory	Description
`/opt/CYRIdiag/j90`	CRAY J90se diagnostic directories and files
`/opt/CYRIlog`	ION and SWS log directory
`/opt/config/topology`	GigaRing definition file
`/opt/config/options`	Boot options file
`/opt/config/sn9###`	Configuration file directory for CRAY J90se mainframe (used in JTAG operations like `mc`)
`/opt/CYRIos/sn9###/param`	UNICOS parameter file
`/opt/CYRIos/sn9###/unicos.###`	UNICOS kernel

sysmon

The `sysmon` command allows you to view, format, and sort the ION log file (`/opt/CYRIlog/ion_syslog`) and SWS log file (`/opt/CYRIlog/sws_syslog`). For more information on the `sysmon` command, refer to the SWS `sysmon` man page or the *SWS-ION Administration and Operations Guide*, publication number SG-2204.

sserrpt

The `sserrpt` command allows you to view MPN error information. You must log into the MPN to use this command. For more information on the `sserrpt` command, refer to the SWS `sserrpt` man page or the *SWS-ION Administration and Operations Guide*, publication number SG-2204.

errpt

The `errpt` command allows you to view UNICOS error information that is stored in `/usr/adm/errfile`. You must log into the mainframe to use this command. For more information on the `errpt` command, refer to the UNICOS `errpt` man page or the *SWS-ION Administration and Operations Guide*, publication number SG-2204.

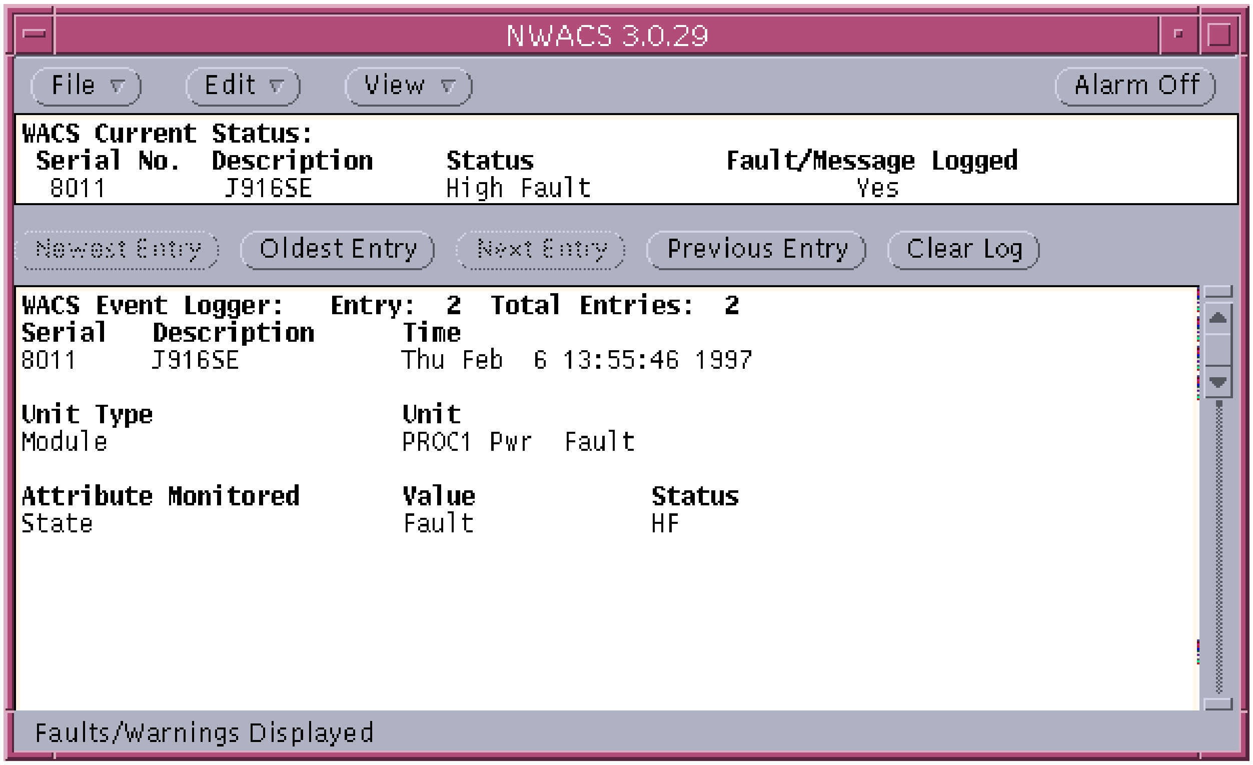
Remote CCU Operations

The central control unit (CCU) for CRAY J90se series systems enables remote monitoring of the system. The RS-232 COMM port on the CCU (connector J4 on CRAY J916se systems and connector J5 on CRAY J932se systems) is connected to one of the serial ports on the SWS. Customers can purchase this enhanced CCU for CRAY J90 series systems.

The NWACS program is used to display the status of the fault LEDs on the CCU, power down the system, reset the mainframe, or reset the I/O. The NWACS program is documented in the *xelog, xcfg, and nwacs User Information* document, Cray Research publication number HDM-012-D.

IMPORTANT! Figure 2 shows the NWACS displays that appear when you are monitoring a remote CRAY J90se CCU. Before switching the CCU from local to remote, be sure to set the 3.3V Margin, 5.0V Margin, and 48VDC switch settings as shown in Figure 2 and click on `Write Switches`. This writes the control settings to the CCU and prevents the system from entering an unpredictable state *or powering down* when you switch to remote.

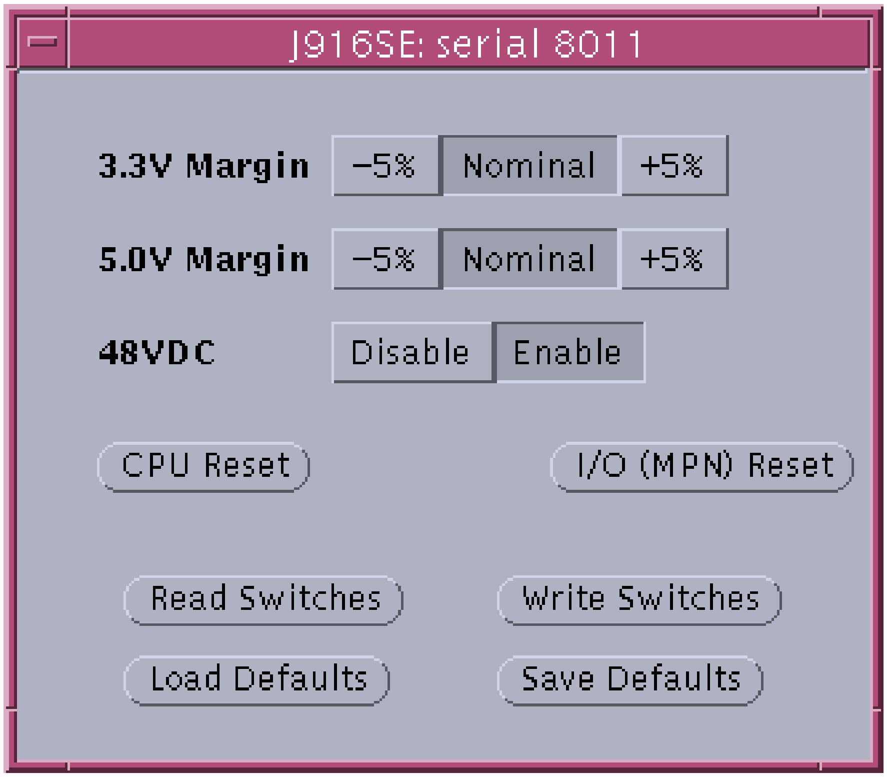
Figure 2. Remote CCU Monitoring with NWACS



```
WACS Current Status:
 Serial No.  Description     Status              Fault/Message Logged
  8011        J916SE         High Fault                    Yes
```

```
WACS Event Logger:   Entry:  2  Total Entries:  2
Serial    Description       Time
8011      J916SE            Thu Feb  6 13:55:46 1997

Unit Type                   Unit
Module                      PROC1 Pwr  Fault

Attribute Monitored         Value            Status
State                       Fault            HF
```

Faults/Warnings Displayed



GigaRing Channel Numbering

Table 2 lists the GigaRing channel numbers for each processor module. One octal GigaRing channel number corresponds to each processor module. The available channel numbers for each system configuration are listed in the UNICOS `param` file. Use these channel numbers when you add GigaRing channels to the system.

Table 2. GigaRing Channel Numbering

Channel Number	Processor Module
024	Processor Module 0
034	Processor Module 1
044	Processor Module 2
054	Processor Module 3
064	Processor Module 4
074	Processor Module 5
104	Processor Module 6
114	Processor Module 7

System-level Troubleshooting

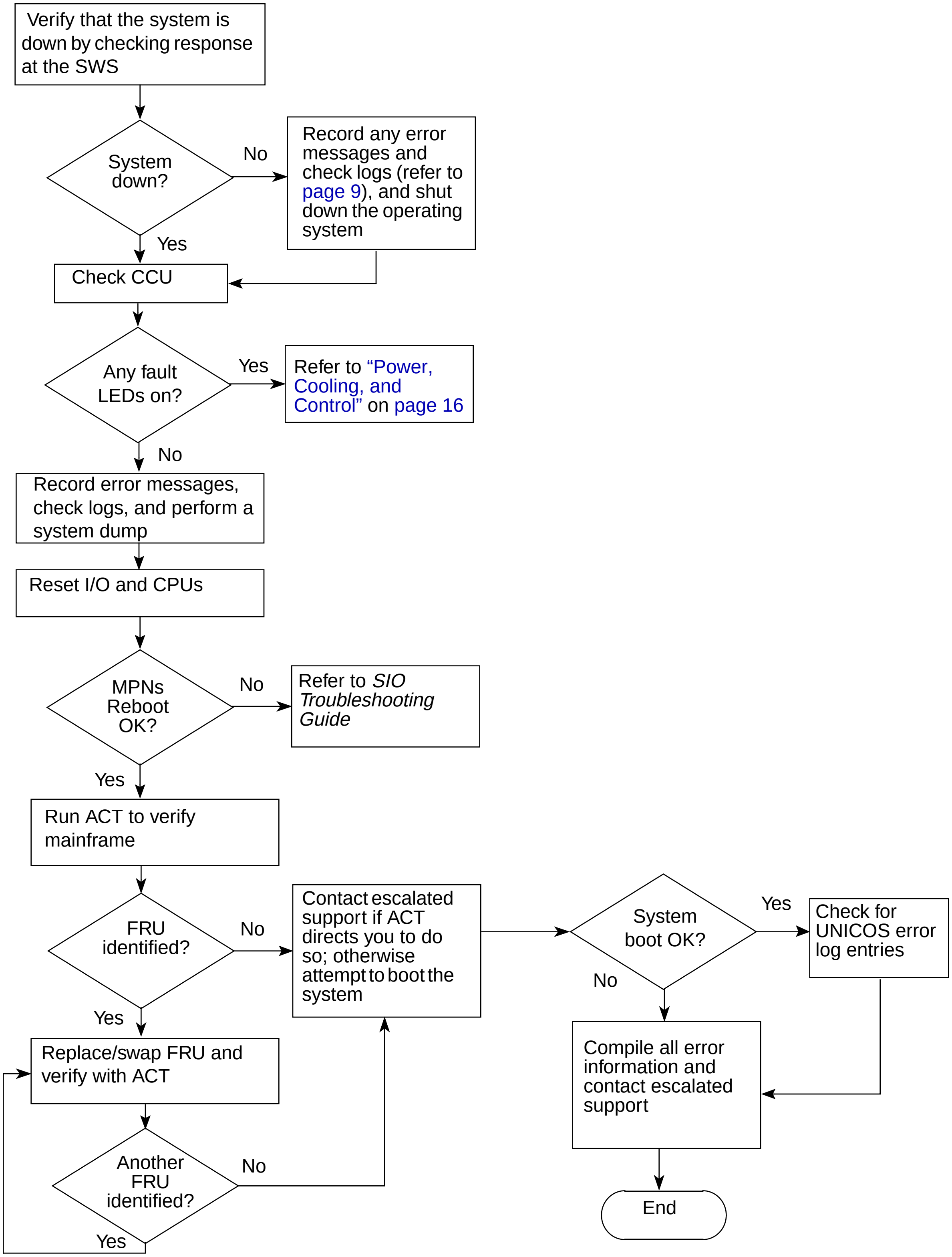
This section provides a system-level troubleshooting flowchart and other troubleshooting procedures. It is organized as follows:

- System-level Troubleshooting Flowchart
- Recording Mainframe Dumps
- System Clock Module Troubleshooting

System-level Troubleshooting Flowchart

The flowchart shown in Figure 3 illustrates the tasks that you should perform when you troubleshoot the system.

Figure 3. System-level Troubleshooting Flowchart


Verify that the system is down by checking response at the SWS
System down?
No
Record any error messages and check logs (refer to page 9), and shut down the operating system
Yes
Check CCU
Any fault LEDs on?
Yes
Refer to "Power, Cooling, and Control" on page 16
No
Record error messages, check logs, and perform a system dump
Reset I/O and CPUs
MPNs Reboot OK?
No
Refer to *SIO Troubleshooting Guide*
Yes
Run ACT to verify mainframe
FRU identified?
No
Contact escalated support if ACT directs you to do so; otherwise attempt to boot the system
System boot OK?
Yes
Check for UNICOS error log entries
No
Compile all error information and contact escalated support
End
Yes
Replace/swap FRU and verify with ACT
Another FRU identified?
No
Yes

Recording Mainframe Dumps

To perform a mainframe dump, use the following `dumpsys` command from the SWS:

```
dumpsys mainframe node name
```

The mainframe node name is listed in the `/opt/config/topology` file on the SWS. The mainframe node name is optional; if a node name is not included with the `dumpsys` command, all the nodes on each ring are dumped, with I/O nodes being dumped first. For more information on the `dumpsys` command, refer to the SWS `dumpsys` man page or the *SWS-ION Administration and Operations Guide*, Cray Research publication number SG-2204.

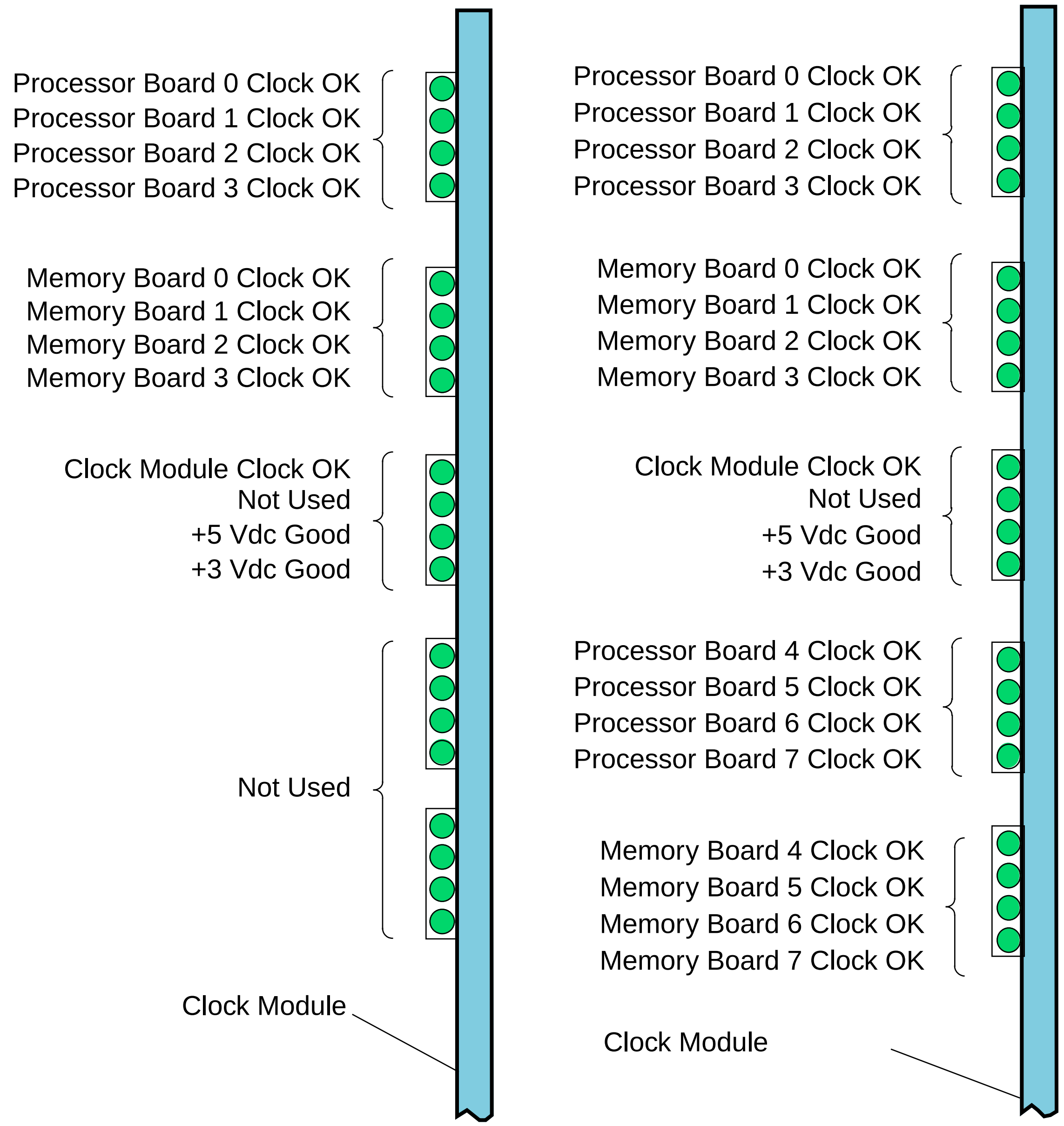
System Clock Module Troubleshooting

If the system is inoperable and you are unable to run any diagnostics, such as `jbs` 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, Cray Research publication number HMM-308-0, 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, (the LED is 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



Clock Module used in a CRAY J98se or CRAY J916se Configuration

Clock Module used in a CRAY J932se Configuration

Mainframe Troubleshooting

A CRAY J90se series mainframe consists of a mainframe cabinet that includes processor, memory, and clock modules; power systems; and control systems. The sections listed below describe the troubleshooting tools necessary for testing and diagnosing failures in the system:

- Power, Cooling, and Control
- Using ACT
- Using JMC

Power, Cooling, and Control

Troubleshooting power, cooling, and control problems requires an understanding of the LED indicators that are provided on the central control unit (CCU) display panel or the NWACS display.

The CCU LEDs provide fault and status indications by using single LED indicators and, in some cases, combinations of LEDs.

NOTE: The N+1 fault LEDs are disabled on all CRAY J90se series systems and CRAY J90 series systems.

The following table provides a troubleshooting chart that includes LED indications, status, the corresponding fault condition or conditions, and corrective actions.

The Alarm column indicates whether the alarm sounds (Yes) or does not sound (No). The Mainframe Status column indicates what action the mainframe takes. The options for this column are:

Status	**Description**
Continues:	Mainframe does not power down or suffer an interrupt
Powers down:	Mainframe stops operation and powers down
Halts:	Mainframe stops operation temporarily until the fault condition is removed

Table 3. Mainframe Power, Cooling, and Control Troubleshooting Chart

LED Indications (an X indicates that the LED is illuminated)																			
PROC TEMP	MEM TEMP	PROC N+1	PROC PWR	MEM N+1	MEM PWR	CLOCK PWR	FAN FAULT	PS FAULT	CNTRL BTRY FAULT	48V CNTRL FAULT	12V CNTRL FAULT	BLOWER FAULT	DC PWR FAULT	AC PWR FAULT	MAINTENANCE MODE	SYSTEM READY	Alarm	Mainframe Status	Fault Condition	Corrective Action
															X		No	Continues	3.3V MARGIN or 5.0V MARGIN switch is not in the NOMINAL position.	Verify that the switches are in the NOMINAL position; correct switch settings if necessary.
																			Battery switch is Off.	Verify that the battery switch is in the On position; correct switch setting if necessary.
																			ALARM ENABLE/INHIBIT switch is in the INHIBIT position.	Verify that each switch is in the ENABLE position; correct switch setting if necessary.
																			3.3V/5.0V ENABLE/INHIBIT switch is in the INHIBIT position.	
																		Powers down	48VDC ENABLE/INHIBIT switch is in the INHIBIT position.	Check the LOCAL/REMOTE switch for REMOTE operation.

Table 3. Mainframe Power, Cooling, and Control Troubleshooting Chart (continued)

LED Indications (an X indicates that the LED is illuminated)																				
PROC TEMP	MEM TEMP	PROC N+1	PROC PWR	MEM N+1	MEM PWR	CLOCK PWR	FAN FAULT	PS FAULT	CNTRL BTRY FAULT	48V CNTRL FAULT	12V CNTRL FAULT	BLOWER FAULT	DC PWR FAULT	AC PWR FAULT	MAINTENANCE MODE	SYSTEM READY	Alarm	Mainframe Status	Fault Condition	Corrective Action
														X			Yes	Powers down	The input AC power has dropped out on one or both lines.	Verify that the customer-supplied power is present and at the nominal voltage. Have customer fix problem if necessary.
																				Verify that the main circuit breaker is closed; close breaker if necessary.
																				Restart mainframe.
													X				Yes	Continues	One of the five 48-Vdc power supplies has a fault condition. (This condition results in a loss of n+1 redundancy.)	Check the 48V status panel to find the power supply with the fault condition; replace the faulty power supply.
			X		X	X				X			X				Yes	Powers down	More than one of the 48-Vdc power supplies has a fault condition.	Check the 48V status panel to find the power supplies with the fault conditions; replace the faulty power supplies.
											X		X				Yes	Continues	12-Vdc power supply is not producing the correct voltage.	Replace front-end power assembly.
															X		No	Power down	LOCAL/REMOTE switch is in the LOCAL position.	Verify that the switch is in the REMOTE position; correct switch setting if necessary.

Table 3. Mainframe Power, Cooling, and Control Troubleshooting Chart (continued)

LED Indications (an X indicates that the LED is illuminated)																				
PROC TEMP	MEM TEMP	PROC N+1	PROC PWR	MEM N+1	MEM PWR	CLOCK PWR	FAN FAULT	PS FAULT	CNTRL BTRY FAULT	48V CNTRL FAULT	12V CNTRL FAULT	BLOWER FAULT	DC PWR FAULT	AC PWR FAULT	MAINTENANCE MODE	SYSTEM READY	Alarm	Mainframe Status	Fault Condition	Corrective Action
									X							X	Yes	Continues	Battery fault because of one of the following conditions:	
																			Battery pack is not connected.	Connect battery pack
																			Battery pack is missing one or more batteries.	Replace missing batteries.
																			One or more of the batteries are shorted.	Replace batteries.

Table 3. Mainframe Power, Cooling, and Control Troubleshooting Chart (continued)

LED Indications (an X indicates that the LED is illuminated)																			
PROC TEMP	MEM TEMP	PROC N+1	PROC PWR	MEM N+1	MEM PWR	CLOCK PWR	FAN FAULT	PS FAULT	CNTRL BTRY FAULT	48V CNTRL FAULT	12V CNTRL FAULT	BLOWER FAULT	DC PWR FAULT	AC PWR FAULT	MAINTENANCE MODE	SYSTEM READY	Alarm	Mainframe Status	Fault Condition	Corrective Action
Numerous faults including a BLOWER FAULT																		Halts	Airflow sensor is not detecting airflow because of one or more of the following conditions:	
																			Blower is not operating at the correct speed.	Verify that the blower switch setting is set at the proper setting for the module configuration. Change setting if necessary.
																			Blower is not receiving power.	Verify that the input voltage is present at the blower output connection on the front-end power assembly.
																				Verify that the blower power cord did not get disconnected at the front-end power assembly and at the blower assembly. Reconnect power cord if necessary.
																			Airflow sensor is faulty.	Replace airflow sensor.
																			Blower motor is not operating correctly.	Replace blower assembly.

Table 3. Mainframe Power, Cooling, and Control Troubleshooting Chart (continued)

LED Indications (an X indicates that the LED is illuminated)																				
PROC TEMP	MEM TEMP	PROC N+1	PROC PWR	MEM N+1	MEM PWR	CLOCK PWR	FAN FAULT	PS FAULT	CNTRL BTRY FAULT	48V CNTRL FAULT	12V CNTRL FAULT	BLOWER FAULT	DC PWR FAULT	AC PWR FAULT	MAINTENANCE MODE	SYSTEM READY	Alarm	Mainframe Status	Fault Condition	Corrective Action
																X	Yes	Continues	Loss of input AC voltage to the PC-10 cabinet.	Refer to the *SIO Troubleshooting Guide* for PC-10 cabinet troubleshooting information.
								X								X	Yes	Continues	The DC output of one or more of the disk peripheral trays has failed.	Refer to the *SIO Troubleshooting Guide* for troubleshooting information.
						X										X	Yes	Continues	Either the 3.3-V and/or the 5.0-V power on the clock module has dropped below acceptable levels.	Replace the module.
					X												Yes	Continues	Either the 3.3-V and/or the 5.0-V power on the corresponding memory module LPM has dropped below acceptable levels. **NOTE:** The module with the faulty LPM halts.	Replace the module.
				X												X	Yes	Continues	One or more of the memory modules' LPM onboard power supplies have failed.	Replace the module.

Table 3. *Mainframe Power, Cooling, and Control Troubleshooting Chart (continued)*

LED Indications (an X indicates that the LED is illuminated)																			
PROC TEMP	MEM TEMP	PROC N+1	PROC PWR	MEM N+1	MEM PWR	CLOCK PWR	FAN FAULT	PS FAULT	CNTRL BTRY FAULT	48V CNTRL FAULT	12V CNTRL FAULT	BLOWER FAULT	DC PWR FAULT	AC PWR FAULT	MAINTENANCE MODE	SYSTEM READY	Alarm	Mainframe Status	Fault Condition	Corrective Action
			X													X	Yes	Continues	Either the 3.3-V and/or the 5.0-V power on the corresponding processor module LPM has dropped below acceptable levels. **NOTE:** The module with the faulty LPM halts.	Replace the module.
	X															X	Yes	Continues	A warning overtemperature condition has occurred in a memory module.	Verify that the blower is operating correctly. Make sure that the filters are not plugged or dirty and that air is flowing freely. Verify that the environmental temperatures are within the correct range.
	X				X												Yes	Powers down	A critical (shut-down) overtemperature condition has occurred in a memory module.	Verify that the blower is operating correctly. Make sure that the filters are not plugged or dirty and that air is flowing freely. Verify that the environmental temperatures are within the correct range.

Table 3. Mainframe Power, Cooling, and Control Troubleshooting Chart (continued)

LED Indications (an X indicates that the LED is illuminated)																Alarm	Mainframe Status	Fault Condition	Corrective Action
PROC TEMP	MEM TEMP	PROC N+1	PROC PWR	MEM N+1	MEM PWR	CLOCK PWR	FAN FAULT	PS FAULT	CNTRL BTRY FAULT	48V CNTRL FAULT	12V CNTRL FAULT	BLOWER FAULT	DC PWR FAULT	AC PWR FAULT	MAINTENANCE MODE	SYSTEM READY			
X																X	Yes	Continues	A warning overtemperature condition has occurred in a processor module.	Verify that the blower is operating correctly. Make sure that the filters are not plugged or dirty and that air is flowing freely. Verify that the environmental temperatures are within the correct range.
X			X															Powers down	A critical overtemperature condition has occurred in a processor module.	Verify that the blower is operating correctly. Make sure that the filters are not plugged or dirty and that air is flowing freely. Verify that the environmental temperatures are within the correct range.

Using ACT

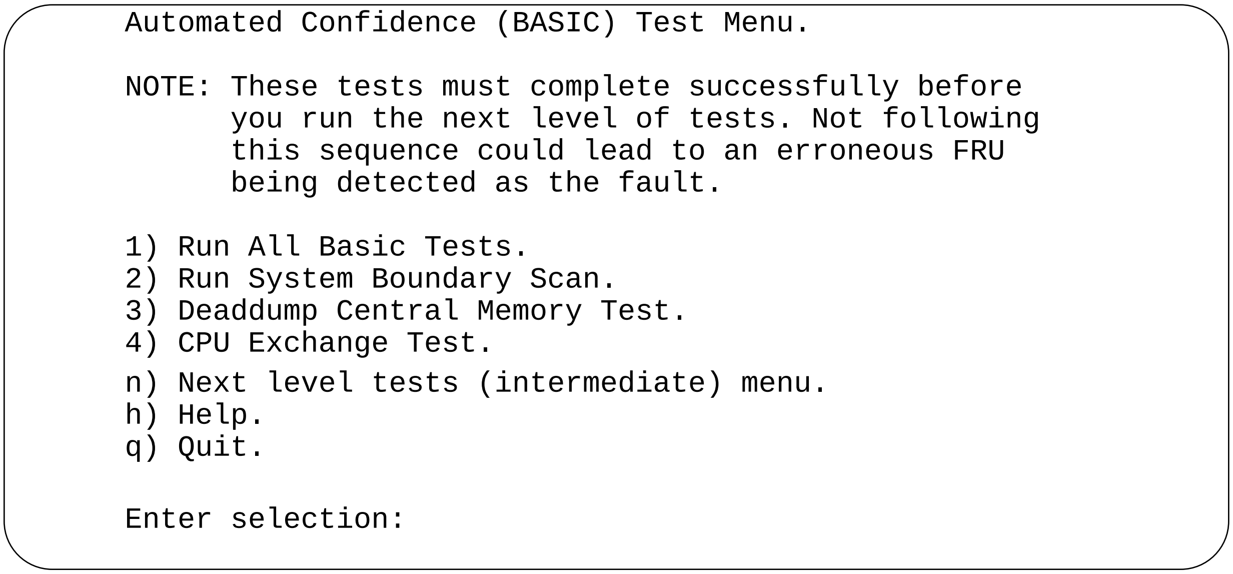
ACT is a menu-driven interface from which the user can select and run offline mainframe diagnostic tests. If you invoke ACT on a system that is running UNICOS, the operating system will crash. Enter `/opt/CYRIdiag/j90/bin/act` to invoke the ACT menu interface.

NOTE: If the `act` command fails, verify that the MPN and GigaRing channel are functional and initialized. To initialize the ring, enter **`bootsys`** *ring name* on the SWS. The correct ring name is listed in the `/opt/config/topology` file and can be identified by the BOOTNODE keyword.

Basic Test Menu

The first menu to appear after you invoke `act` is the basic test menu. The tests that are available from the basic menu are designed to verify basic functionality of the system. You should verify that all the tests on the basic menu ran successfully before you proceed to the intermediate level tests. Figure 5 shows the basic test menu.

Figure 5. ACT Basic Test Menu



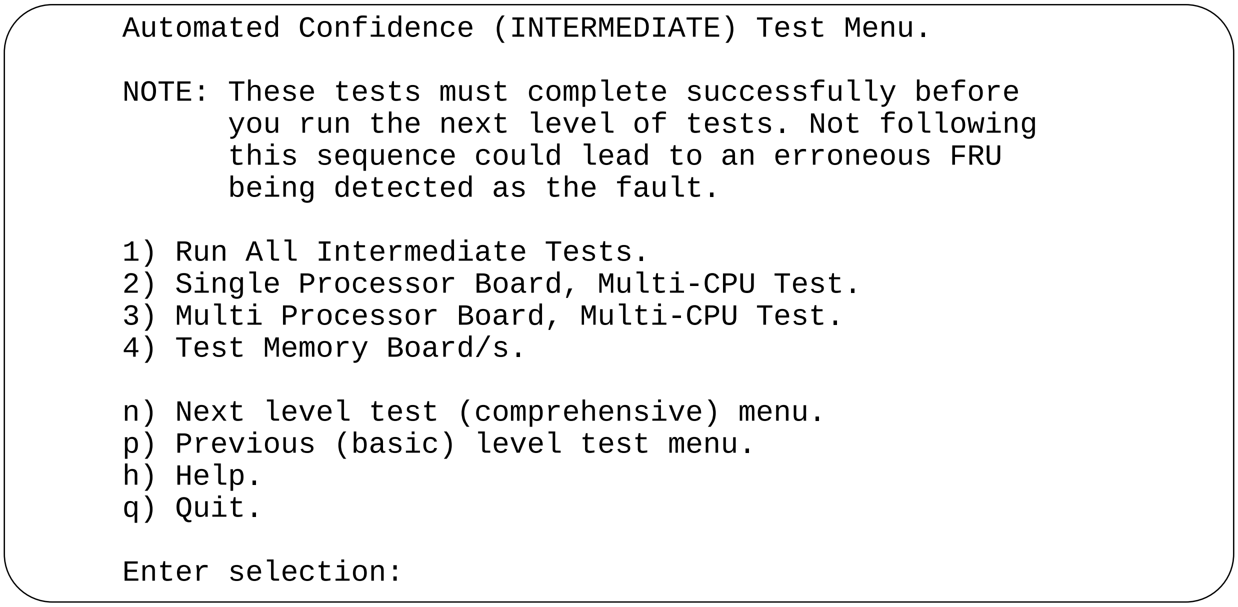
```
Automated Confidence (BASIC) Test Menu.

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 Tests.
2) Run System Boundary Scan.
3) Deaddump Central Memory Test.
4) CPU Exchange Test.
n) Next level tests (intermediate) menu.
h) Help.
q) Quit.

Enter selection:
```

- Select 1 to run all basic tests. If a failure is detected, a FRU isolation routine begins. If a FRU is isolated, the test stops, displays the failing FRU, and logs the failure in /opt/CYRIdiag/j90/tmp/act_err.log. If a failure is detected but the failing unit cannot be identified, the test informs the user to contact escalated support.
- Select 2 to run the boundary scan test jbs. Refer to "Using JBS" for more information on jbs.
- Select 3 to run the deaddump mainframe memory test. This test checks whether mainframe memory can be written to and read from. The user can enter a beginning and ending address if desired.
- Select 4 to test the CPU exchange mechanism. This test checks whether the exchange mechanism is functional in each of the configured CPUs. The exchange mechanism assigns control of a CPU from one program to another.
- Select n to go to the ACT intermediate test menu (refer to Figure 6).
- Select h for help.
- Select q to quit the ACT menu system.

Intermediate Test Menu

The second level in the ACT menu system is the intermediate test menu; refer to Figure 6. The diagnostic tests that are available from the intermediate level are designed to run on a system that has some functionality. Intermediate diagnostics should be run only after all tests from the basic menu have completed successfully.

Figure 6. ACT Intermediate Test Menu



```
Automated Confidence (INTERMEDIATE) 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 Intermediate Tests.
2) Single Processor Board, Multi-CPU Test.
3) Multi Processor Board, Multi-CPU Test.
4) Test Memory Board/s.

n) Next level test (comprehensive) menu.
p) Previous (basic) level test menu.
h) Help.
q) Quit.

Enter selection:
```

- Select `1` to run all intermediate tests. This selection runs all the offline mainframe diagnostics from selections `2` through `4`.

- Select `2` to test a single processor board. This selection enables you to select a particular processor board or a combination of processor boards for testing. The tests from this selection verify that the CPUs are functional and are limited to the CPUs on an individual processor board. Table 5 describes each diagnostic.

- Select `3` to test multiple processor boards. This selection enables you to select a particular processor board or a combination of processor boards for testing. The tests from this selection verify that the CPUs and inter-CPU communications are functional. Unlike the previous selection, these tests are not limited to the CPUs on an individual processor board. Table 5 describes each diagnostic.

- Select `4` to test mainframe memory board(s). This selection enables you to select a particular processor board or a combination of processor boards to test memory operations.Table 5 describes each diagnostic.

- Select `n` to go to the ACT comprehensive test menu (refer to Figure 7).

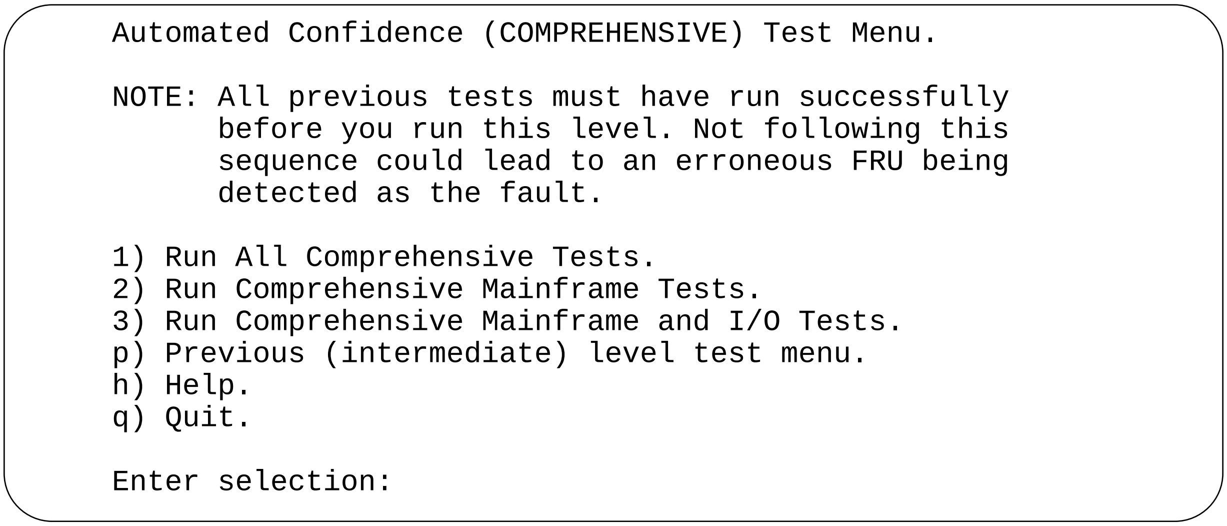
- Select `h` for help.

- Select `q` to quit the ACT menu system.

Comprehensive Test Menu

The third level in the ACT menu is the comprehensive menu; refer to Figure 7. The diagnostic tests that are available from the comprehensive level are confidence tests that verify system functionality. Comprehensive diagnostics should be run only after all tests from the basic and intermediate menus have completed successfully.

Figure 7. ACT Comprehensive Test Menu



```
Automated Confidence (COMPREHENSIVE) Test Menu.

NOTE: All previous tests must have run successfully
      before you run this level. Not following this
      sequence could lead to an erroneous FRU being
      detected as the fault.

1) Run All Comprehensive Tests.
2) Run Comprehensive Mainframe Tests.
3) Run Comprehensive Mainframe and I/O Tests.
p) Previous (intermediate) level test menu.
h) Help.
q) Quit.

Enter selection:
```

- Select `1` to run all comprehensive tests. This selection runs all the diagnostics from menu selections `2` through `3`.
- Select `2` to run comprehensive mainframe tests. This selection enables you to select a particular processor board or a combination of processor boards for testing. The tests from this selection run random CPU and shared register instructions that are designed to verify the overall functionality of the mainframe.
- Select `3` to run comprehensive mainframe and I/O tests. This selection enables you to select a particular processor board or a combination of processor boards for testing. This selection runs a CRAY J90se client test and a GigaRing test to verify mainframe to GigaRing communications.
- Select `p` to return to the ACT intermediate test menu.
- Select `h` for help.
- Select `q` to quit the ACT menu system.

Failure Information

If ACT detects a failure, it runs a FRU analyzer subroutine to identify the failing unit. This subroutine lists possible failing units in descending order, with the most likely FRU on the top of the list and the least likely at the bottom. Figure 8 shows an example of ACT fault output.

The FRU analyzer is not always 100% accurate; the list of FRUs should be interpreted as a list of suspected components that provide the user with a starting point in the troubleshooting process. If a failure is detected but the FRU cannot be identified, ACT displays a standard message that instructs the user to contact escalated support.

Figure 8. ACT Error Example

```
jbtas test in progress.....................done
jsr2 test in progress......................timeout error
Checking for memory errors...none found
Appending failure to tmp/act_err.log...done

***** FAULT DETECTED *****
Error Code Received = 77000000 Hex
CPU Error Code Received = 00F00000 Hex

PROBABLE FRU LIST
        CPU 24, Processor Module 5
        CPU 25, Processor Module 5
        CPU 26, Processor Module 5
        CPU 27, Processor Module 5
Press [CR] to continue.
```

NOTE: The error codes displayed from an ACT fault are used by escalated support to help locate code within the ACT program. The error codes are considered irrelevant for field use.

Using JBS

The CRAY J90se series boundary scan (JBS) test checks the mainframe module interconnections for shorts and opens. All of the boundary flip-flops on each ASIC are connected to form a serial scan chain. Test patterns are shifted into each ASIC. One system clock is applied and the test pattern is shifted out. By checking the response, data faults such as opens and shorts can be detected at the board (module) level and across the backplane.

NOTE: For JBS to run properly, the board revision level under JBS must match the boundary scan number (BSN) revision level of the modules being tested. The BSN revision level is located on a label that is affixed to the faceplate of each module.

The boundary scan test is normally used from ACT (under basic menu item 1 or 2); JBS can perform the following tests:

- Scan chain integrity test
- Board-level boundary scan test
- Backplane-level boundary scan test

Boundary Scan Sequence

If JBS runs in default mode, (ACT basic menu item 1), then no user intervention is required. The application performs the following functions:

1. Reads configuration files to determine the backplane type, number of boards in the system, board boundary scan revisions, and other information. If the menu system is invoked (ACT basic menu item 2), the user is prompted to change the test parameters.

2. Performs a scan chain integrity test on each board that is selected for testing. This test verifies that the boundary scan chain is intact and operational. If an error is encountered during the test, the failing boards are called out and no further tests are performed.

3. Performs a board-level boundary scan test on each board that is selected for testing. This test verifies the interconnections of the on-module interconnect network. If an error is encountered during this test, a header is displayed that describes the error format, followed by a line for each failure that is detected. Multiple failures may be displayed for a single board.

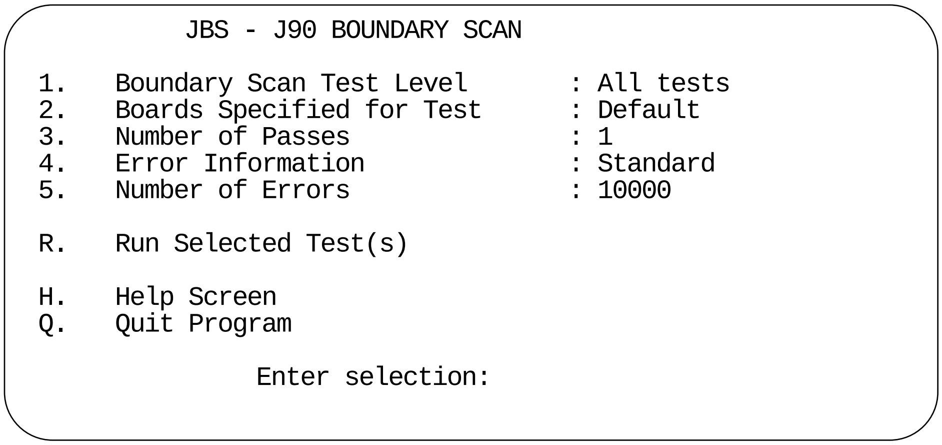
4. Performs a backplane-level boundary scan test. This test verifies the interconnections of the off-module interconnect network (networks that pass through the backplane). If an error is encountered during this test, the failing network information is displayed.

5. Displays the pass/fail results to the screen and saves them to the `/opt/CYRIdiag/J90/jbs.log` file.

Running the Test

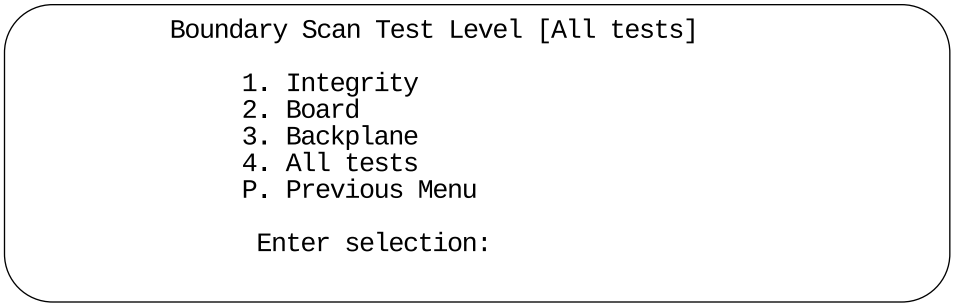
The following paragraphs provide a brief tutorial for running the boundary scan test using the menu system.

1. Enter **`/opt/CYRIdiag/j90/bin/jbs -menu`** to display the main JBS menu. The main JBS menu enables you to run the boundary scan test on a specific board configuration. The following screen snap shows the JBS main menu:



```
            JBS - J90 BOUNDARY SCAN

1.   Boundary Scan Test Level       : All tests
2.   Boards Specified for Test      : Default
3.   Number of Passes               : 1
4.   Error Information              : Standard
5.   Number of Errors               : 10000

R.   Run Selected Test(s)

H.   Help Screen
Q.   Quit Program

                 Enter selection:
```

2. Select 1 from the main menu to choose a board-level or system-level test. From this menu, you have the following options:



```
           Boundary Scan Test Level [All tests]

                1. Integrity
                2. Board
                3. Backplane
                4. All tests
                P. Previous Menu

                 Enter selection:
```

Select **1** to verify that the boundary scan chain is intact and operational. If an error is encountered during the test, the failing boards are displayed and no further tests are performed.

Select **2** to verify the interconnections of the on-module interconnect network. If an error is encountered during this test, a header that describes the error format is displayed, followed by a line for each detected failure. Multiple failures may be displayed for a single board.

Select **3** to verify the interconnections of the off-module interconnect network (networks that pass through the backplane). If an error is encountered during this test, the failing network information is displayed.

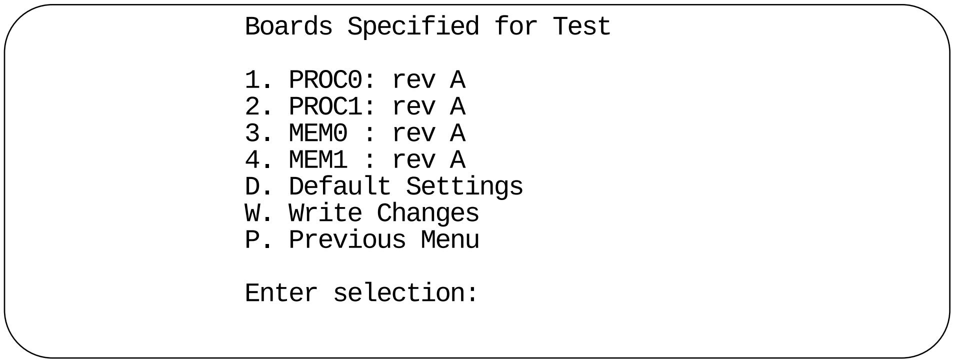
Select **4** to run all the tests.

Select **P** to return to the previous (main) menu.

NOTE: The setting shown in brackets at the top of the menu is the default setting.

3. Return to the main menu and then select **2** to display specific boards to test (as shown below). This menu enables you to specify which boards you want to test and also displays the boundary scan revision level for each board.

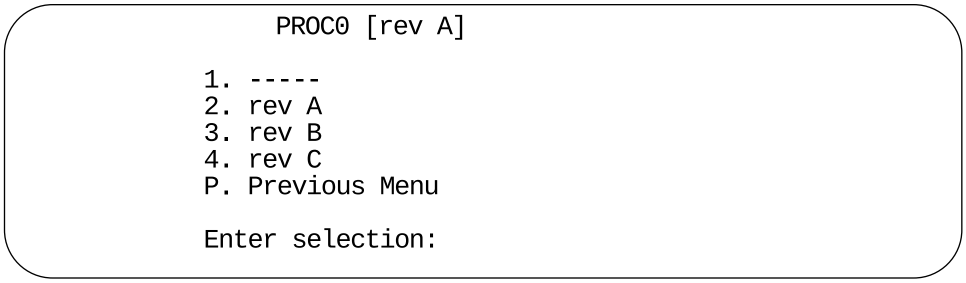
NOTE: For JBS to run properly, the board revision level under JBS must match the boundary scan number (BSN) revision level of the modules being tested. The BSN revision level is located on a label that is affixed to the faceplate of each module.



```
Boards Specified for Test

1. PROC0: rev A
2. PROC1: rev A
3. MEM0 : rev A
4. MEM1 : rev A
D. Default Settings
W. Write Changes
P. Previous Menu

Enter selection:
```

NOTE: If “-----” displays after the board designator, it means that the board will not be tested.

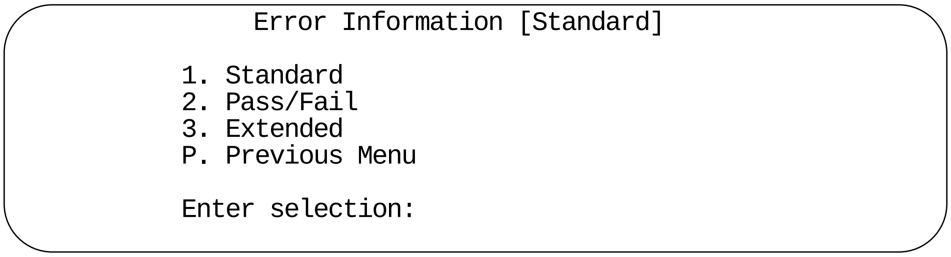
4. Select **1** from the Boards Specified for Test menu to display the various revision levels for a specific board. The following menu shows the revision levels for the MEM0 board.



```
                 PROC0 [rev A]

            1. -----
            2. rev A
            3. rev B
            4. rev C
            P. Previous Menu

            Enter selection:
```

5. Return to the main menu and select **3** to alter the number of passes that the test will take. Enter a new value from 0 to 100 to change the pass value.

6. Return to the main menu and select **4** to control the type of error information that is output from the test. The following menu is displayed.



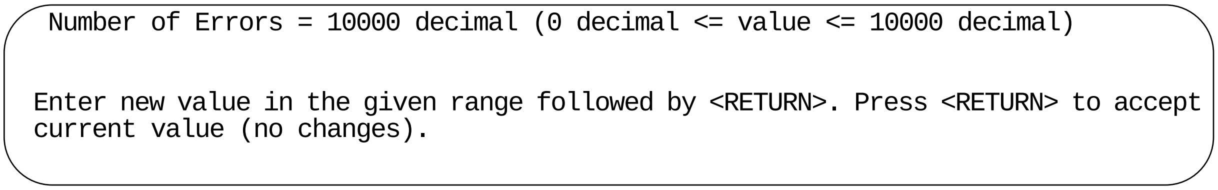
```
                 Error Information [Standard]

            1. Standard
            2. Pass/Fail
            3. Extended
            P. Previous Menu

            Enter selection:
```

Select **1** to display the standard error information.

Select **2** to display only the pass/fail information.

Select **3** to display the same information as standard and to generate a file with the specific vector number and expected data for each failure.

7. Return to the main menu and select **5** to display the following menu. Enter a new value, or press return, to return to the main menu.



```
 Number of Errors = 10000 decimal (0 decimal <= value <= 10000 decimal)


Enter new value in the given range followed by <RETURN>. Press <RETURN> to accept
current value (no changes).
```

8. Select **R** from the main menu to run the selected test.

9. Select **H** from the main menu to display help information.

10. Select **Q** from the main menu to quit.

Error Information

The following paragraphs describe the different types of failures that JBS detects.

Integrity Test Failures

The integrity test is not a boundary scan test. It verifies that the boundary scan chain is intact and functional. If the integrity test fails, be sure that you are testing the correct board. In addition, ensure that the board you are testing has power applied. If the integrity test fails, the scan chain is broken, and you must replace the board.

Onboard Net Failure

If a specific board failure is detected, a header displays that describes the error format and is followed by a line for each failure that is detected. Multiple failures may be displayed for a single board.

Backplane Net Failure

A specific board failure cannot be detected because the failure could be on the source board, the backplane, or the destination board. A header displays that describes the error format and is followed by a line for each failure that is detected.

When a backplane failure is encountered, you should document the failing boundary scan positions, error masks, board types, and their boundary scan revision numbers on the form that is provided with the board.

From this information, Central Repair can determine which chip or pin failed and which specific test patterns caused the failure. This enables Central Repair to duplicate the error and verify that the failure was corrected before the board is sent to another site.

The boundary scan chain position and error mask can also be used to isolate backplane level failures (instead of using the entire net).

The backplane failure information is displayed in the following format:

```
Performing backplane level boundary scan test...failed.

Board   BScan Error     Net: <driver> <connector> <connector> <receiver>
Name    Pos.   Mask           Failing location in parenthesis; [*problem]
------ ----- -----     --------------------------------------------------
MEM0    02057 015E615F PROC1_U4/A406 J2/AB01 J5/K01 (MEM0_U33/A014) *SA1

Completed 001 pass(s); copy of results stored in ”jbs.log”.
```

- `Board Name.` Name of the board that detected the failure.

- `BScan Pos.` Boundary scan chain position on board; used to determine net information. Send this information to Central Repair so the failure can be verified and corrected.

- `Error Mask.` Mask of boundary scan patterns that the node failed. This should be sent to Central Repair so the failure can be verified and corrected (this explains exactly which patterns of the test set failed).

- `Net:` The net contains four nodes: The first node listed is the driving node (output pin); the second and third nodes are the backplane connectors that the net passes through. The fourth node is the receiver (input pin). The specific node that detected the failure is in parentheses.

- `*problem.` This field will contain `*SA0`, `*SA1`, or will not be present.

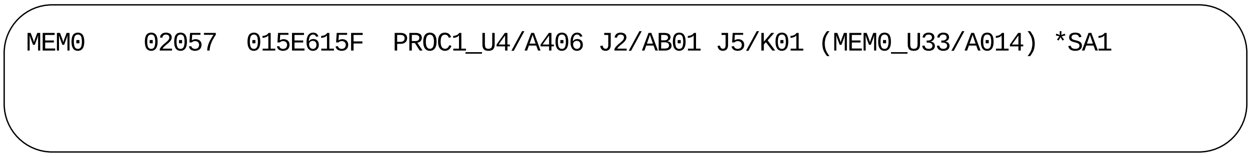
  `*SA0` - If the net fails all patterns in the set that have the value of 1, then this field contains the string *SA0 for stuck-at-zero. If a net is stuck-at-zero, then it may be open, shorted to ground, or shorted to a net that is not being tested.

  `*SA1` - If the net fails all patterns in the set that have the value of 0, then this field contains the string `*SA1` for stuck-at-one. If a net is stuck-at-one, then it may be open or shorted to power.

  Not present (empty). If the net failed some patterns with a value of one, and/or some patterns with a value of zero, it could be shorted to another net that is being tested (more than one net will have failures). Not present could also mean that an intermittent problem (short or open) exists.

Backplane Failures

In order to isolate a backplane failure using boundary scan, boards must be moved to determine whether the failure is in the backplane or on the board. The following screen snap shows an example backplane failure. Use the following procedure to isolate a backplane failure.



```
MEM0    02057  015E615F  PROC1_U4/A406 J2/AB01 J5/K01 (MEM0_U33/A014) *SA1
```

1. Swap MEM0 with another memory board and re-run the test.

   If the same failure is displayed, then the problem is in either the backplane or the PROC1 board. Proceed to Step 2.

   If a different failure is displayed, the problem is on the board that you moved. This failure display starts with a new board name (for example, MEM1 if the board that was originally in MEM0 was moved to MEM1).

2. Swap PROC1 board with another processor board and re-run the test a third time. If the same failure is displayed, then the problem is in the backplane.

   If a different failure is displayed, the failure is on the board that was moved. This failure display starts with a new board name, and this board should be replaced.

   NOTE: The names MEM0, MEM1, PROC1, etc. are fixed, depending on the backplane, and do not change if boards are switched (they represent the board types of each slot).

Examples

The following screen snaps are examples of the boundary scan display. The example in Figure 9 shows that the test is passing with no failures.

Figure 9. Boundary Scan Test - No Failures

```
JBS - J90 Boundary Scan application (Tue Feb  4 08:18:34 1997)

JBS Configuration: system=bp2x2
     The following boards are selected for test:
          PROC0 rev A
          PROC1 rev A
          MEM0  rev A
          MEM1  rev A

     Please verify the boards selected for test, and the boundary
        scan revision of each board (must match BSN label on module).
     Run jbs with the “-menu” option to reconfigure.

Generating boundary scan vectors for backplane level test...please wait...

*Tue Feb  4 08:18:50 1997 - starting pass 001/001

Performing scan chain integrity test...passed.

Performing board level boundary scan test...passed.

Performing backplane level boundary scan test...passed.

JBS Completed 001 pass(s); copy of results stored in “jbs.log”.
```

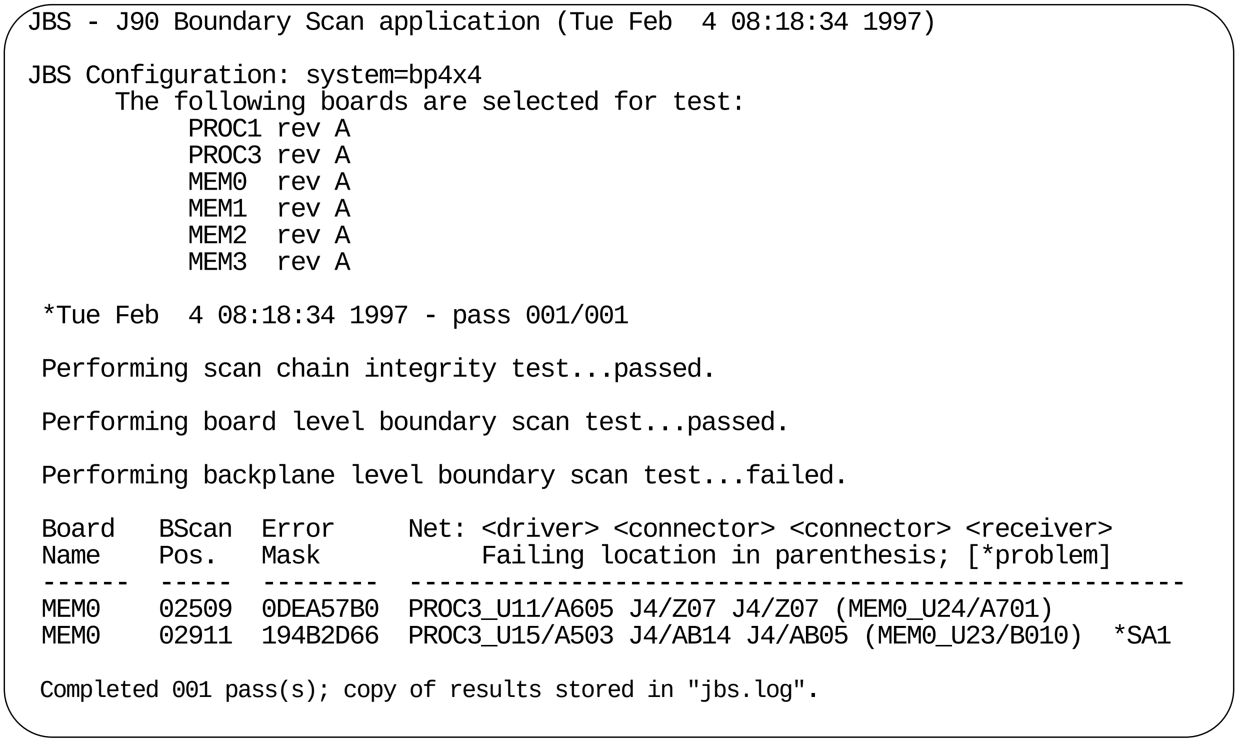
The example in Figure 10 shows that the integrity test is failing. You must replace MEM0, MEM1, and PROC1 and rerun the test.

Figure 10. Boundary Scan Test - Integrity Test Failure

```
JBS - J90 Boundary Scan application (Tue Feb  4 08:18:34 1997)

JBS Configuration: system=bp4x4
     The following boards are selected for test:
          PROC0 rev A
          PROC1 rev A
          PROC2 rev A
          PROC3 rev A
          MEM0  rev A
          MEM1  rev A
          MEM2  rev A
          MEM3  rev A

*Tue Feb  4 08:18:34 1997 - pass 001/001

Performing scan chain integrity test...failed.
     Errors were encountered on the following board(s):
          MEM0
          MEM1
          PROC1

Unable to perform boundary scan tests because of failures.

CPU=00>
```

The example in Figure 11 shows the information that is displayed when a backplane net failure occurs and the steps that were taken to isolate the problem.

Figure 11. Backplane Net Failure - First Pass



```
JBS - J90 Boundary Scan application (Tue Feb  4 08:18:34 1997)

JBS Configuration: system=bp4x4
      The following boards are selected for test:
           PROC1 rev A
           PROC3 rev A
           MEM0  rev A
           MEM1  rev A
           MEM2  rev A
           MEM3  rev A

 *Tue Feb  4 08:18:34 1997 - pass 001/001

 Performing scan chain integrity test...passed.

 Performing board level boundary scan test...passed.

 Performing backplane level boundary scan test...failed.

 Board   BScan  Error     Net: <driver> <connector> <connector> <receiver>
 Name    Pos.   Mask           Failing location in parenthesis; [*problem]
 ------  -----  --------  -----------------------------------------------------
 MEM0    02509  0DEA57B0  PROC3_U11/A605 J4/Z07 J4/Z07 (MEM0_U24/A701)
 MEM0    02911  194B2D66  PROC3_U15/A503 J4/AB14 J4/AB05 (MEM0_U23/B010)  *SA1

 Completed 001 pass(s); copy of results stored in "jbs.log".
```

There are two backplane failures from PROC3 to MEM0. These failures could be on either board or in the backplane. In order to isolate the failures, you must move either PROC3 or MEM0 and rerun the test. In this example the MEM0 and MEM2 boards were swapped. Figure 12 shows the error information that was displayed after the MEM0 and MEM2 boards were swapped.

The failures moved from MEM0 to MEM2, so we know that these failures are on the memory module that is currently in MEM2. If the failures remained with MEM0, then they are on either PROC3 or the backplane.

It is possible that the two failures may reside in different locations. For example, one failure may be on the memory board and another on the processor board. When troubleshooting, diagnose each failure separately.

Figure 12. Backplane Net Failure - Second Pass

```
JBS - J90 Boundary Scan application (Tue Feb  4 08:18:34 1997)

JBS Configuration: system=bp4x4
       The following boards are selected for test:
            PROC1 rev A
            PROC3 rev 0
            MEM0  rev A
            MEM1  rev A
            MEM2  rev A
            MEM3  rev A

  *Tue Feb  4 08:18:34 1997- pass 001/001

  Performing scan chain integrity test...passed.

  Performing board level boundary scan test...passed.

  Performing backplane level boundary scan test...failed.

  Board   BScan  Error     Net: <driver> <connector> <connector> <receiver>
  Name    Pos.   Mask           Failing location in parenthesis; [*problem]
  ------  -----  --------  ---------------------------------------------------
  MEM2    02509  029C3940  PROC3_U11/A410 J4/Y02 J4/Z07 (MEM2_U24/A701)
  MEM2    02911  19BD4266  PROC3_U15/A009 J4/AB11 J4/AB05 (MEM2_U23/B010)  *SA1

  Completed 001 pass(s); copy of results stored in "jbs.log".
```

Using JMC

The JMC application is a graphical user interface that is used to troubleshoot a CRAY J90se series mainframe. JMC was designed for use by Cray Research Systems Test and Checkout (STCO) personnel but it can also be used by service engineers. JMC should be considered an advanced tool because it offers more flexibility and capability than ACT but is more difficult to use and does not identify any failing FRUs.

NOTE: If you invoke JMC on a system that is running UNICOS, the operating system will crash.

Enter **`/opt/home/crayadm/jmc`** to invoke the JMC window.

Commonly Used JMC Commands

Table 4 summarizes the commonly used JMC commands for troubleshooting. The following "JMC Command Reference" section includes a complete list of JMC commands. These commands are invoked from the JMC interface window.

Table 4. Commonly Used JMC Commands

Command	Notes
`cls`	Clears the JMC window.
`dm <address>`	Displays a single mainframe memory location.
`dmm <address>`	Creates a real-time mainframe memory display window. Default window displays 4 quadrants of memory. Refer to Figure 13.
`ds`	Deadstarts a program in mainframe memory.
`exit/quit`	Exits JMC.
`fm`	Fills a series of memory locations with a data pattern.
`help`	Displays all JMC commands.
`jbs`	Refer to the "Using JBS" section, page 29 (usually used from ACT).
`jconfig`	Refer to "Using the jconfig Utility" section, page 62 (`jconfig` can be entered directly from the SWS command line).
`mc`	Master clear initialization of mainframe (halts any running processes).
`offline`	Loads offline diagnostics (refer to the "Running Offline Diagnostics" section, page 50).
`stat`	Displays mainframe CPU status (refer to the "Using the stat Command" section, page 61).

JMC Command Reference

The following list of commands can be used from the JMC window. Refer to Table 4 for a list of commonly used commands.

`act`

Starts the ACT utility.

NOTE: Running `act` from within the JMC window can cause the GigaRing channel to hang. Run `act` from a shell window and not from the JMC window.

`adrm <h|o|d>`

Sets the default input mode to hexidecimal, octal, or decimal.

`am <addr[-prcl] <[pa] [pb] [pc] [pd]`

Address modify. Sets the mainframe memory address (parcel A through parcel D). Entering a comma does not alter the parcel. The `am` command functions in the same way as the `set` command.

`atm <addr> [pa] [pb]`

Modifies a word in TIB memory.

`ar`

Displays the system scan window.

`autocon [on | off]`

Displays or sets autocon mode. If set to on, a `console on` command is automatically invoked when a command is entered that requires the use of the conbus.

`bpc [cpu]`

Disables a CPU breakpoint trigger and clears PMATCH.

`bpd [cpu]`

Disables a CPU breakpoint.

`bpe [cpu]`

Enables a CPU breakpoint.

`bpl`

Lists breakpoint information for all slots and CPUs.

`bpm <mem> <mar #> <mcmatch>`

Sets a MEM MCMATCH register (always triggered).

`bpp [cpu] <pmatch[-p]>`

Sets a CPU PMATCH register and trigger.

`call <label>`

Script command. Calls a function. Function must start with a label and end with a return.

`cbt <fn> <slot>`

Scans `<slot>` and compares it to the data in `<fn>`.

`cfg`

Reads `.cfg` files for all mapped slots.

`chipmap [slot]`

Displays a slot’s chip bitmap. With no arguments, all of the slot maps are displayed; if a slot number is specified, that slot’s chip map is displayed.

`clk <on | off | n>`

Turns system clock on or off, or steps `n` times (65535 max).

`cls`

Clears the screen.

`clb`

Clears the screen and the JMC text window buffer.

`cmd_echo <on | off>`

Enables and disables command echo in command scripts.

`cmt <fn>`

Compares memory to a text file and reports errors.

`console <on|off>`

Turns system conbus on or off.

`count <init | inc | dec | print>`

Manipulates the script counter in command scripts.

`cpu <cpu #>`

Sets current default CPU number and prompt.

`date`

Displays current date and time.

`dd [cpu | kill]`

Runs deaddump loop. Enter `dd kill` to stop the loop.

`debug <on|bin|pkt|off>`

Turns debug mode on or off.

`derr <addr>`

Displays 20 words of formatted mainframe memory error information at `<address>`.

`dm <addr> [count]`

Displays a snapshot of mainframe memory at `<address>`. The count defaults to 16.

`dmc <addr> [count]`

Displays mainframe memory at `<address>` in Cray Assembly Language (CAL). The count defaults to 16.

`dmm [uladdr] [uraddr] [lladdr] [lraddr]`

Displays the memory quad window. Addresses can be entered in each quadrant.

`ds [fn] [cpu]`

Deadstarts `[cpu]` (default = 0). Loads a file `[fn]` if entered.

`dtm <addr> [count]`

Displays client TIB memory at address. Count defaults to 16.

```
dxp [addr]
```

Displays static exchange package at address `[addr {0}]`.

```
echo [message]
```

Echoes `[message]` to console. Used for command scripts.

```
exit
```

Quits `jmc`.

```
exmod <addr>
<preg|iba|ila|dba|dla|xa|vl|cln|modes|a0-7|s0-7> <value>
```

Modifies exchange package field at `<addr>` with `<value>`. `<value>` is a single word except for registers S0 through S7, which is in parcel format (up to 4 parcels).

```
fm <addr> <count> <[pa] [pb] [pc] [pd]>
```

Fills memory with parcel A through parcel D. Entering a comma clears that parcel. Entering no parcel options fills memory with an address pattern.

```
ftm <addr> <count> <[pa] [pb]>
```

Fills TIB RAM with parcel A through parcel B. A comma clears that parcel. Specifying no parcels fills TIB RAM with an address pattern.

```
goto <label>
```

Script command. Jumps to `<label>` in a command script.

```
help [command]
```

Prints command reference.

```
if <value> goto <label>
```

Script command. Checks current status against `<value>`. If equal, control jumps to the line after `<label>` in the command script.

```
ifnot <value> goto <label>
```

Script command. Checks current status against `<value>`. If not equal, control jumps to the line after `<label>` in the command script.

```
ifge <value> <goto|call> <label>
```

Script command. Checks current status against `<value>`. If status is greater than or equal to `<value>`, control jumps to the line after `<label>` in the script.

```
ifle <value> [goto|call] <label>
```

Script command. Checks current status against `<value>`. If status is greater than or equal to `<value>`, control jumps to the line after `<label>` in command script.

```
iu <on|off> [cpu]
```

Enables or disables CPU instruction unit.

```
jbs
```

Runs boundary scan. This is normally invoked from ACT.

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

Starts the `jconfig` utility. This command can also be entered from a shell window.

```
jtag reset
```

Resets all test access port (TAP) controllers in the system.

```
jtag check <slot>
```

Performs JTAG sanity check and reports errors.

```
jtag report <slot>
```

Similar to `jtag check`, but it reports all information.

```
jtag config <slot> <file>
```

Loads JTAG config registers into `<slot>` from `<file>`.

```
jtag dc <slot> [file]
```

Displays JTAG configuration data.

`jtag shadow <slot> [file]`

Reads JTAG shadow registers from `<slot>` into `[file]`.

`jtag id <slot> [file]`

Reads JTAG IDCODE registers from `<slot>` into `[file]`.

`jtag ir <slot> [file]`

Reads JTAG IR registers from `<slot>` into `[file]`.

`jtag preload <in | out> <slot> <file>`

Preloads JTAG boundary scan information to/from `<file>`.

`jtag extest <in | out> <slot> <file>`

Preloads EXTEST JTAG Boundary scan information to/from `<file>`.

`lbt <fn> <slot>`

Performs an internal scan into slot `<slot>` from file `<fn>`.

`ld <fn> [addr]`

Loads binary file `<fn>` into memory at `[addr]`. If no address is specified, the image is loaded at address 0.

`lmt <fn>`

Loads text file into memory.

`loop`

Displays a low-level loop control window.

`mc`

Master clears the system.

`mem0 <start> <end> <patt> [op] [mask]`

I/O based Memory Test.

`<start>`: Starting Address

`<end>`: Ending Address

`<patt>`: Data Pattern (1 = address pattern, 2 = all 0's, 4 = all 1's, 10 = odd bits, 20 = even bits, 40 = random)

`<op>`: Operation. Default = Do not Loop On Error, R/W (6). 1 = loop on error, 2 = write only, 4 = read only, 10 = random start and length, within `<start>` and `<end>`, 20 = similar to as 10 but uses the same random seed number.

`<mask>`: data mask. Default = 1's. 0 bits ignored. If mask is 0, no compare is performed.

`mlog`

Displays the memory error log window.

`mm <start> <count> [prcl 0]... [prcl 3]`

Searches memory for pattern match.

`msg [message]`

Echoes `[message]` to console (for command scripts).

`offline [-c #] [-n #] [-b #] [-d] [-k monitor] [-e] [-m #] [-l #] [-s #] <fn>`

Loads, configures, and attaches monitor to the specific offline diagnostic.

-c: Octal bitmask selection of CPUs to test. (for example `-c 0377`)

`-n`: Octal bitmask selection of CPUN to test. (for example `-n 0377`)

`-b`: Octal bitmask selection of banks to test. (for example `-b 01000`)

`-d`: Disable cache. (default is defined within `jconfig`)

`-k`: Monitor mode: NONE, YMM, YMI, YMS, YSMI, YM8.)

`-e`: Enable cache. (default is defined within jconfig)

`-m`: Last memory address in octal. (for example `-m 177777777` for 32 MWords)

`-l`: Number of clusters in octal. (for example `-l 0,..., 41`)

`-s`: Octal bitmask section(s) select. (for example `-s 17`)

`parcel <number>`

Converts raw number into address-parcel format.

`pll [slot]`

Reports the PLL LOCKED bits of each ASIC in `[slot]`. If `[slot]` is omitted, all mapped boards are reported.

`quit`

Quits `jmc`.

`return`

Returns from a command script function (refer to `call`).

`reset <on|off>`

Sets or clears the system RESET line.

`rupt [cpu]`

Sends an MCU interrupt to current CPU or <cpu>.

`s <addr>[-prcl] <[pa][-prcl]... [pn][-prcl]>`

Sets word or parcel(s) word(s).

`sa <fn> [start[-p]] [end]`

Saves memory from [start {0}] to [end {start + 16}].

`sh <command> [args]`

Executes a UNIX command or shell. `<command>` cannot be interactive (for example, it does not work with the `vi` editor).

`sbt <fn> <slot>`

Performs an internal scan from slot `<slot>` into file `<fn>`.

`script_exit [rcode]`

Exits a CMD SCRIPT with [rcode{default = 0}].

`script_pause`

Pauses a command script. Pressing return resumes the script.

`smt <fn> [start] [cnt]`

Saves [cnt {16}] words of memory from address[start {0}] into a TEXT file.

`spvt`

Displays the supervisory channel test window.

`stat`

Displays the updating CPU status window.

`statf <slot>`

Displays all Status Register bits for `<slot>` once.

`stati`

Displays status on command window once.

`step [n] [cpu]`

Steps [n {1}] instructions on [CPU {current}].

`sysmap`

Displays Processor Board map, CPU map, and MEM map.

`test <clk | p> [cpu]`

Script command. Returns current value of !STOPCLK or PREG.

`time`

Displays current date and time.

`tml <addr>`

Returns the lower 32 bits from memory address `<addr>`.

`tmlu <addr>`

Returns the upper 32 bits from memory address `<addr>`.

`update <on | off>`

Turns the automatic update feature for dynamic windows on or off.

`verbose <on | off>`

Sets/clears verbose mode.

`wait <secs>`

Script command. Waits `<secs>` seconds before continuing.

`wpc <addr> <count> [err addr]`

Script command. Waits until value in `<addr>` is greater than or equal to `<count>` or until value in `[err addr]` is nonzero.

`wpcc <cpumask> <count> [f]`

Script command. Waits until smallest pass count of all CPUs in `<cpumask>` is greater than or equal to `<count>`, or optionally until any CPU’s fail count is nonzero.

Running Offline Diagnostics

The `offline` command provides a means to load and deadstart offline diagnostics and utilities. This section shows you how to use offline diagnostics to troubleshoot the mainframe.

Offline Diagnostics Quick Reference

Table 5 provides a quick reference of the CRAY J90se series mainframe diagnostic descriptions. Offline diagnostic listings are available on the Service Publications and Training Web site at `http://servinfo.cray.com/j90`.

Table 5. Mainframe Offline Diagnostic Descriptions

Name	Test Description
initmmr	Initializes the GigaRing node MMRs
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
jcrid	Multiple-CPU, random-instruction test
jdump	Dumps B, T, SB, ST, semaphore and vector registers to memory
jejt	Exchange jump test
jexch	Exchange package functions utility
jexd	Cause an exchange and hang
jfind	Finds a user entered data pattern
jfpt	Floating-point units test
jibt	Instruction buffer test
jiot	Comprehensive multi-CPU based I/O test
jjpt	Jump test
jmnum	Writes each memory location with its address.
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

Table 5. Mainframe Offline Diagnostic Descriptions (continued)

Name	Test Description
jscl	Scalar logical basic test
jscs	Scalar shift test
jsct	Scalar cache test
jscx	Mainframe based GigaRing and Client 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 6 lists the mainframe offline diagnostics by function to help you determine which diagnostics to use to check out areas of the mainframe.

Table 6. 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	jscx

Using the Offline Command

Escalated support personnel can use the `offline` command in conjunction with JMC 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 is invoked from within JMC, and it obtains configuration information from the system configuration files. The `offline` command options enable you to specify a monitor, a particular set of CPUs, memory size, etc. Use the `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 `dmm` (display mainframe memory), `ds` (deadstart), and `mc` (master clear) commands. You should also have access to the offline diagnostic listings, which are available on the Service Publications and Training Web site at:

`http://servinfo.cray.com/j90`

Enter the following commands to set up the JMC window so you can view the mainframe memory standard diagnostic memory locations. The standard diagnostic memory locations are listed in Table 7 on page 54.

1. Enter **`jmc`** to display the JMC window.
2. Enter `adrm o` if you would like to display octal instead of hexadecimal.
3. Enter `mc` to master clear the system.
4. Enter `offline jclr`.
5. Enter `ds` to deadstart `jclr` and clear the system.
6. Enter **`dmm`** to display the mainframe memory window.
7. Enter a memory address of 1100 in the upper-left quadrant of the mainframe memory window and an address of 1120 in the lower-left quadrant of the mainframe memory window.
8. In the upper-right quadrant, use the right mouse button to change the mode to exchange package. This is set to memory location 5000 for the master CPU.

9. In the lower-right quadrant of the memory window, enter an address of 1700 to display the memory error table.

10. Enter `mc` to clear the system.

11. In the JMC window, enter **`offline`** *testname* to load a test.

12. In the JMC window, enter **`ds`** to start the test.

NOTE: Always clear memory as described on page 56 before you run another diagnostic.

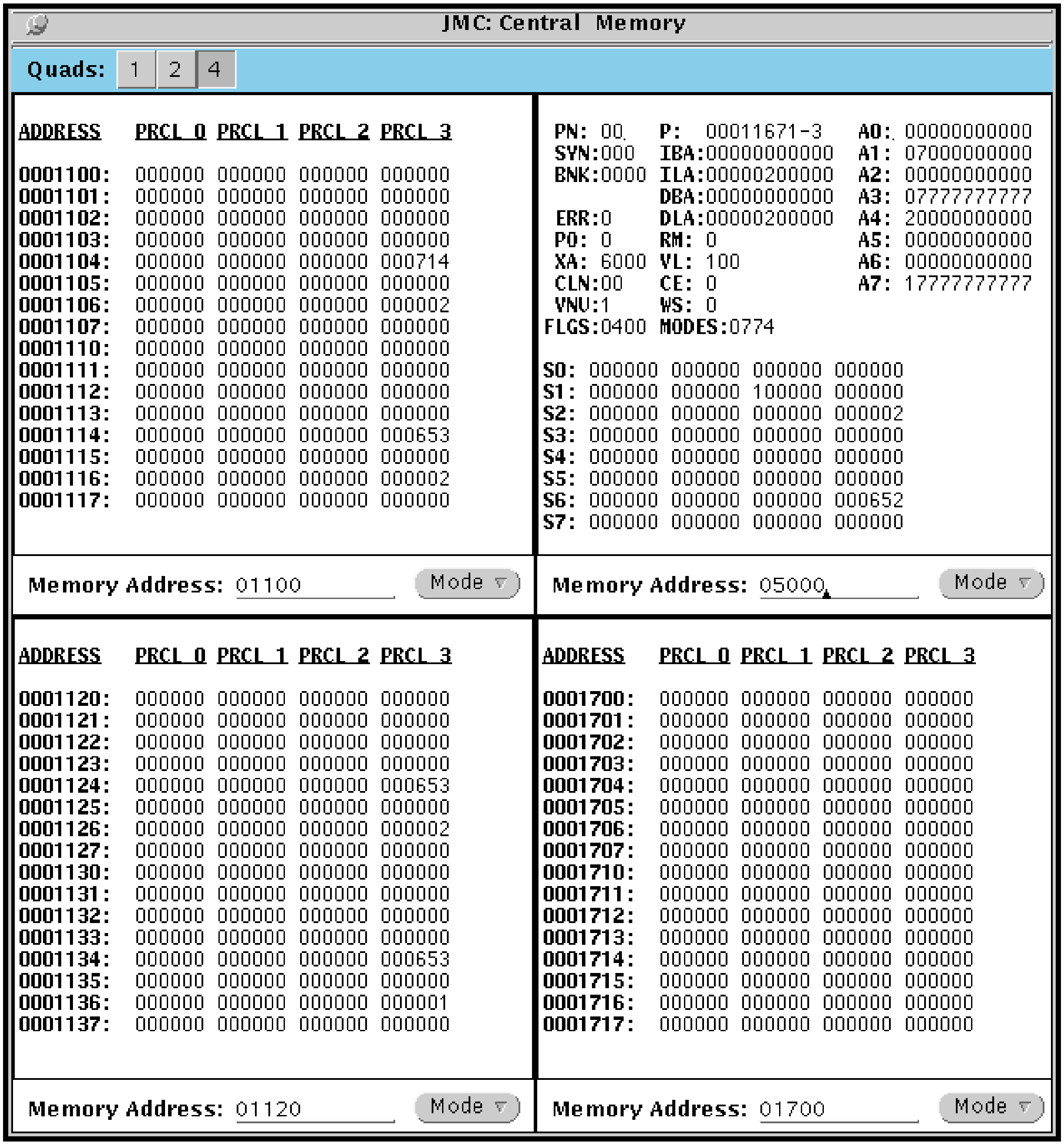
Table 7 lists the standard memory locations.

Table 7. Standard Diagnostic 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 – 1107	Open
1110 – 1117	CPU 1
1120 – 1127	CPU 2
.	
.	
.	
1270 – 1277	CPU 15
5000 – 5017	CPU 0 or master exchange package
5020 – 5760	CPUs 1 - 31 current exchange package
1700 – 1777	Memory error table (default)
2000	Start of test code

Figure 13 shows the `dmm` window after these commands have been entered.

Figure 13. Offline Diagnostics Output



```
ADDRESS    PRCL 0 PRCL 1 PRCL 2 PRCL 3

0001100:   000000 000000 000000 000000
0001101:   000000 000000 000000 000000
0001102:   000000 000000 000000 000000
0001103:   000000 000000 000000 000000
0001104:   000000 000000 000000 000714
0001105:   000000 000000 000000 000000
0001106:   000000 000000 000000 000002
0001107:   000000 000000 000000 000000
0001110:   000000 000000 000000 000000
0001111:   000000 000000 000000 000000
0001112:   000000 000000 000000 000000
0001113:   000000 000000 000000 000000
0001114:   000000 000000 000000 000653
0001115:   000000 000000 000000 000000
0001116:   000000 000000 000000 000002
0001117:   000000 000000 000000 000000

Memory Address: 01100
```

```
 PN: 00    P:   00011671-3   A0: 00000000000
 SYN:000   IBA:00000000000   A1: 07000000000
 BNK:0000  ILA:00000200000   A2: 00000000000
           DBA:00000000000   A3: 07777777777
 ERR:0     DLA:00000200000   A4: 20000000000
 P0: 0     RM: 0             A5: 00000000000
 XA: 6000  VL: 100           A6: 00000000000
 CLN:00    CE: 0             A7: 17777777777
 VNU:1     WS: 0
FLGS:0400  MODES:0774

S0: 000000 000000 000000 000000
S1: 000000 000000 100000 000000
S2: 000000 000000 000000 000002
S3: 000000 000000 000000 000000
S4: 000000 000000 000000 000000
S5: 000000 000000 000000 000000
S6: 000000 000000 000000 000652
S7: 000000 000000 000000 000000

Memory Address: 05000
```

```
ADDRESS    PRCL 0 PRCL 1 PRCL 2 PRCL 3

0001120:   000000 000000 000000 000000
0001121:   000000 000000 000000 000000
0001122:   000000 000000 000000 000000
0001123:   000000 000000 000000 000000
0001124:   000000 000000 000000 000653
0001125:   000000 000000 000000 000000
0001126:   000000 000000 000000 000002
0001127:   000000 000000 000000 000000
0001130:   000000 000000 000000 000000
0001131:   000000 000000 000000 000000
0001132:   000000 000000 000000 000000
0001133:   000000 000000 000000 000000
0001134:   000000 000000 000000 000653
0001135:   000000 000000 000000 000000
0001136:   000000 000000 000000 000001
0001137:   000000 000000 000000 000000

Memory Address: 01120
```

```
ADDRESS    PRCL 0 PRCL 1 PRCL 2 PRCL 3

0001700:   000000 000000 000000 000000
0001701:   000000 000000 000000 000000
0001702:   000000 000000 000000 000000
0001703:   000000 000000 000000 000000
0001704:   000000 000000 000000 000000
0001705:   000000 000000 000000 000000
0001706:   000000 000000 000000 000000
0001707:   000000 000000 000000 000000
0001710:   000000 000000 000000 000000
0001711:   000000 000000 000000 000000
0001712:   000000 000000 000000 000000
0001713:   000000 000000 000000 000000
0001714:   000000 000000 000000 000000
0001715:   000000 000000 000000 000000
0001716:   000000 000000 000000 000000
0001717:   000000 000000 000000 000000

Memory Address: 01700
```

Offline Command Examples

The following subsections provide examples of how you can use the `offline` command.

NOTE: All commands are entered from a JMC window.

Clearing Memory

You must clear memory with the `jclr` utility before you run any tests on mainframe memory. The following example shows the `jclr` utility loaded and deadstarted with the `offline` and `ds` commands. The `dmm` command is used to view the CPU pass counters so you can verify that `jclr` has completed one pass.

```
CPU=00> mc
CPU=00> offline jclr
CPU=00> ds
CPU=00> dmm 1100
CPU=00> mc
```

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 following example, the `jaab` diagnostic will execute in all CPUs. The `stat` command will show that all the CPU P registers are incrementing.

```
CPU=00> (refer to the previous subsection "Clearing Memory")
CPU=00> mc
CPU=00> offline jaab
CPU=00> ds
CPU=00> stat (displays the CPU status)
```

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). The pass count for CPU 1 is displayed at memory location 1104, and the exchange package is displayed at memory location 5000.

```
CPU=00> (refer to the previous subsection "Clearing Memory")
CPU=00> mc
CPU=00> offline jaab
CPU=00> ds 1
```

Viewing CPU Exchange Packages

To view an exchange package, use one of the memory display quadrants on a `dmm` window. Change the memory location to the memory address that contains the exchange package for that CPU (refer to Table 7 on page 54) and select the exchange package mode with the mode button.

Refer to the *CRAY J90 Series System Programmer Reference*, Cray Research publication number CSM-0301-0A0, for more information about the exchange package.

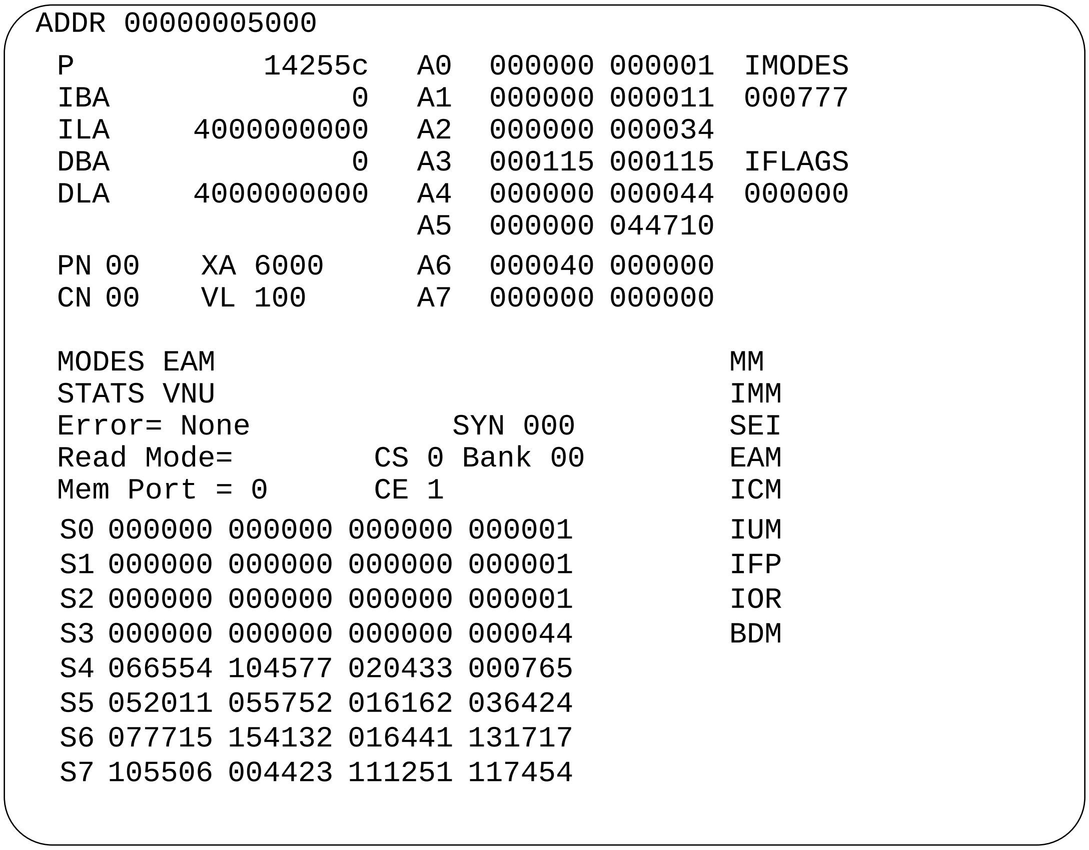
CPU 1 and CPU 0 change places in the exchange package area of memory. The master CPU always uses memory location 5000. The `PN` (processor number) field should equal 1 because of the `ds 1` command that was issued earlier.

Adjust the `dmm` display to show the exchange package at location 5020. The `PN` field should equal 0.

CPU 3 is the master CPU in the following example. 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.

```
CPU=00> (refer to the previous subsection “Clearing Memory”)
CPU=00> mc
CPU=00> offline jaab
CPU=00> ds 3
```

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



```
ADDR 00000005000
 P           14255c   A0  000000 000001  IMODES
 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

CPU=00> (refer to the previous subsection "Clearing Memory")
CPU=00> **mc**
CPU=00> **offline -c 371 jaab**
CPU=00> **ds**
CPU=00> **stat** *(do not enter if stat display (is already up)*

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 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

```
CPU=00> (refer to the previous subsection “Clearing Memory”)
CPU=00> mc
CPU=00> offline -c 341 jaab
CPU=00> ds 4
CPU=00> stat  do not enter if stat display (page 61) is already up
```

Displaying Memory Error Information

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

```
    CPU=00> dm 1700

Parcel   0 1 2 3
1700      134508   137073   000531   050076
Parcel 1 = 137073 (1 011 111 000 111 011)
```

```
EM PORT RM      BIT       BANK   BANK      Proc.No. Syndrome  Count
00  000 00x  x0000000  xxxxxx00 00000000  xxx00000 00000000  16 bits
```

Parcel 0 Parcel 1 Parcel 2 Parcel 3

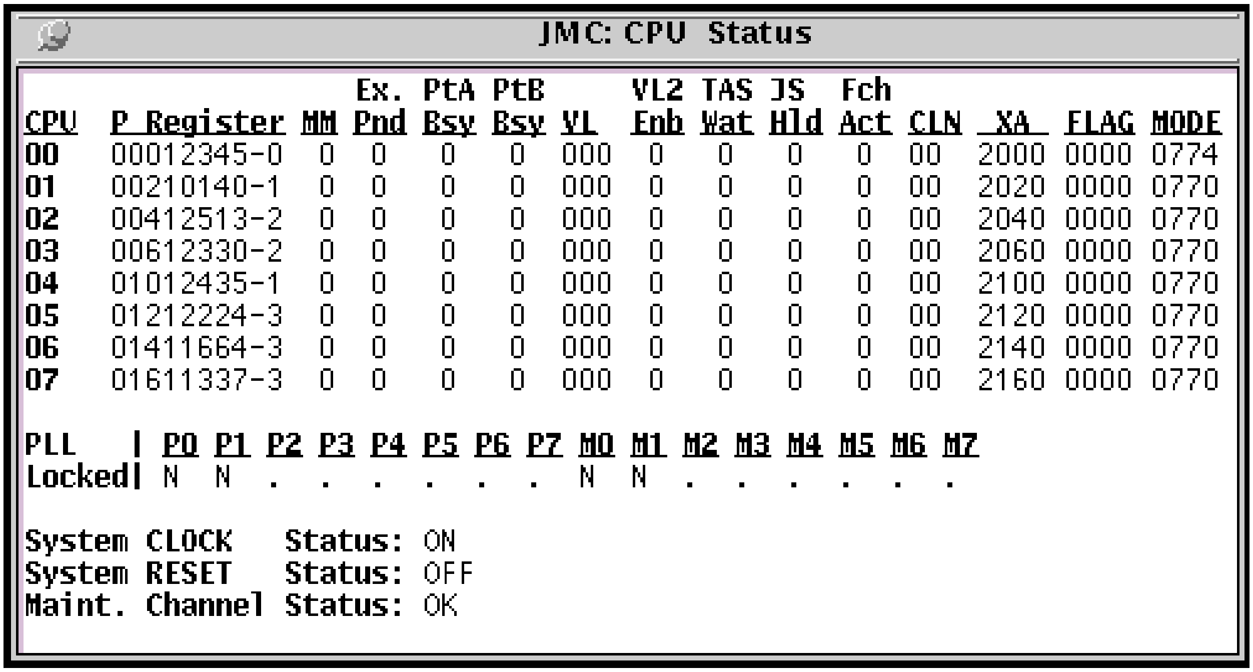
Count	-	Total count	Bits	0 – 15
Syndrome	-	Syndrome number	Bits	16 – 23
Proc.No.	-	Processor number	Bits	24 – 28
	-	Unused	Bits	29 – 31
BANK	-	Bank	Bits	32 – 41
	-	Unused	Bits	42 – 47
BIT	-	Bit number	Bits	48 – 54
	-	Unused	Bits	55 – 56
RM	-	Read mode	Bits	57 – 58
PORT	-	Port	Bits	59 – 61
E	-	Error mode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

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 as shown in Figure 14.

Enter the `stat` command in the JMC window while offline diagnostics are running to view the activity of each CPU.

Figure 14. Display Window for stat Command



```
                          Ex. PtA PtB      VL2 TAS JS  Fch
CPU   P Register   MM Pnd Bsy Bsy VL   Enb Wat Hld Act CLN  XA   FLAG MODE
00    00012345-0   0  0   0   0   000  0   0   0   0   00  2000 0000 0774
01    00210140-1   0  0   0   0   000  0   0   0   0   00  2020 0000 0770
02    00412513-2   0  0   0   0   000  0   0   0   0   00  2040 0000 0770
03    00612330-2   0  0   0   0   000  0   0   0   0   00  2060 0000 0770
04    01012435-1   0  0   0   0   000  0   0   0   0   00  2100 0000 0770
05    01212224-3   0  0   0   0   000  0   0   0   0   00  2120 0000 0770
06    01411664-3   0  0   0   0   000  0   0   0   0   00  2140 0000 0770
07    01611337-3   0  0   0   0   000  0   0   0   0   00  2160 0000 0770

PLL   | P0 P1 P2 P3 P4 P5 P6 P7 M0 M1 M2 M3 M4 M5 M6 M7
Locked| N  N  .  .  .  .  .  .  N  N  .  .  .  .  .  .

System CLOCK   Status: ON
System RESET   Status: OFF
Maint. Channel Status: OK
```

Using the jconfig Utility

NOTE: If you are unfamiliar with the `jconfig` utility, you can corrupt the system configuration files and make the system inoperable. This tool should be used only by qualified support personnel.

Enter **`/opt/CYRIdiag/j90/bin/jconfig`** to start the jconfig utility. Refer to the man page on `jconfig` for a description of all the command line options.

The `jconfig` utility (refer to Figure 16), enables you to build and edit CRAY J90se 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 `/opt/config/sn9###/pm`*[0-7]*`.cfg` for processor modules and `/opt/config/sn9###/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 that is needed by the configuration registers and 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`.

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.

jconfig Sequence

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

1. Resets the CRAY J90se series system so that it can perform maintenance functions. Any processes that were running in the system are halted; `jconfig` will crash UNICOS if it is running.

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 using an information screen that you may 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.

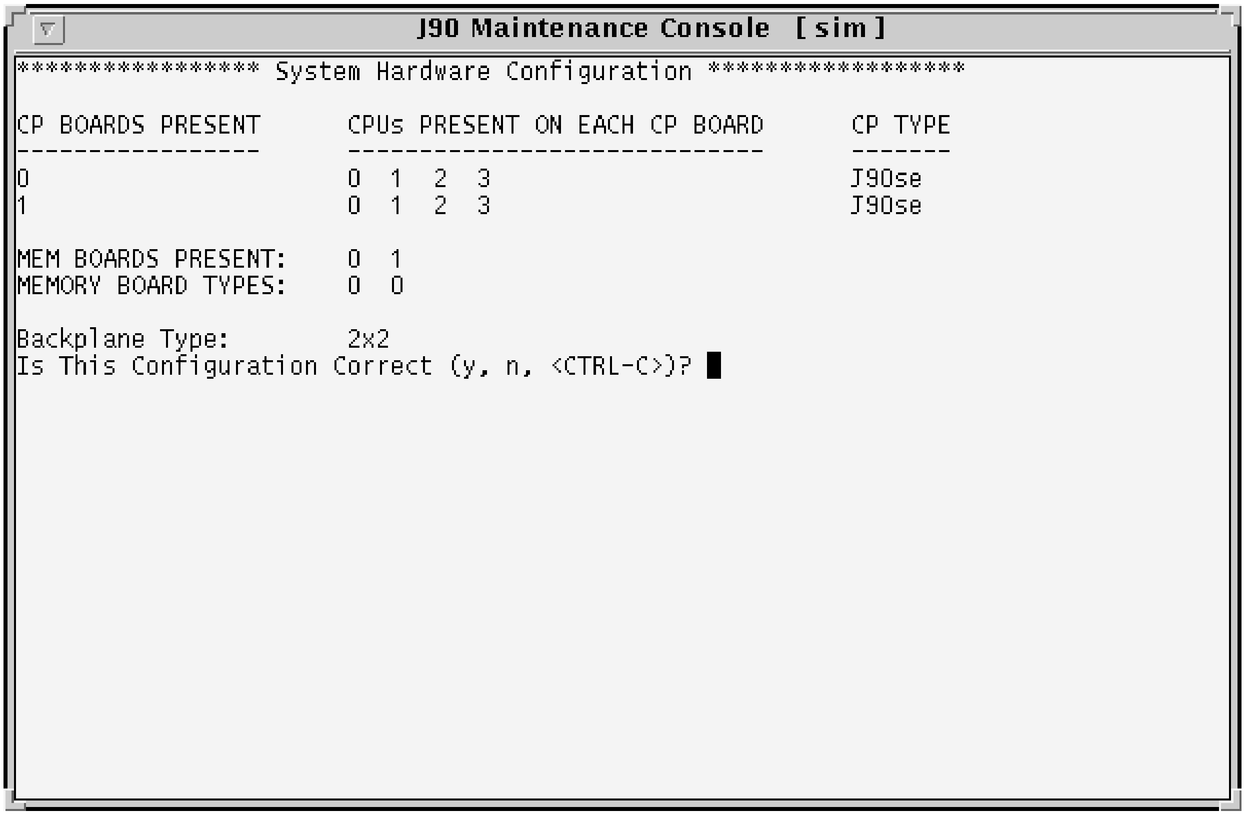
Configuration Information

The `jconfig` utility requires hardware configuration information such as which CPU slots have processor modules in them (refer to Figure 15 and Figure 16), 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 GigaRing channel 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 located, `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.

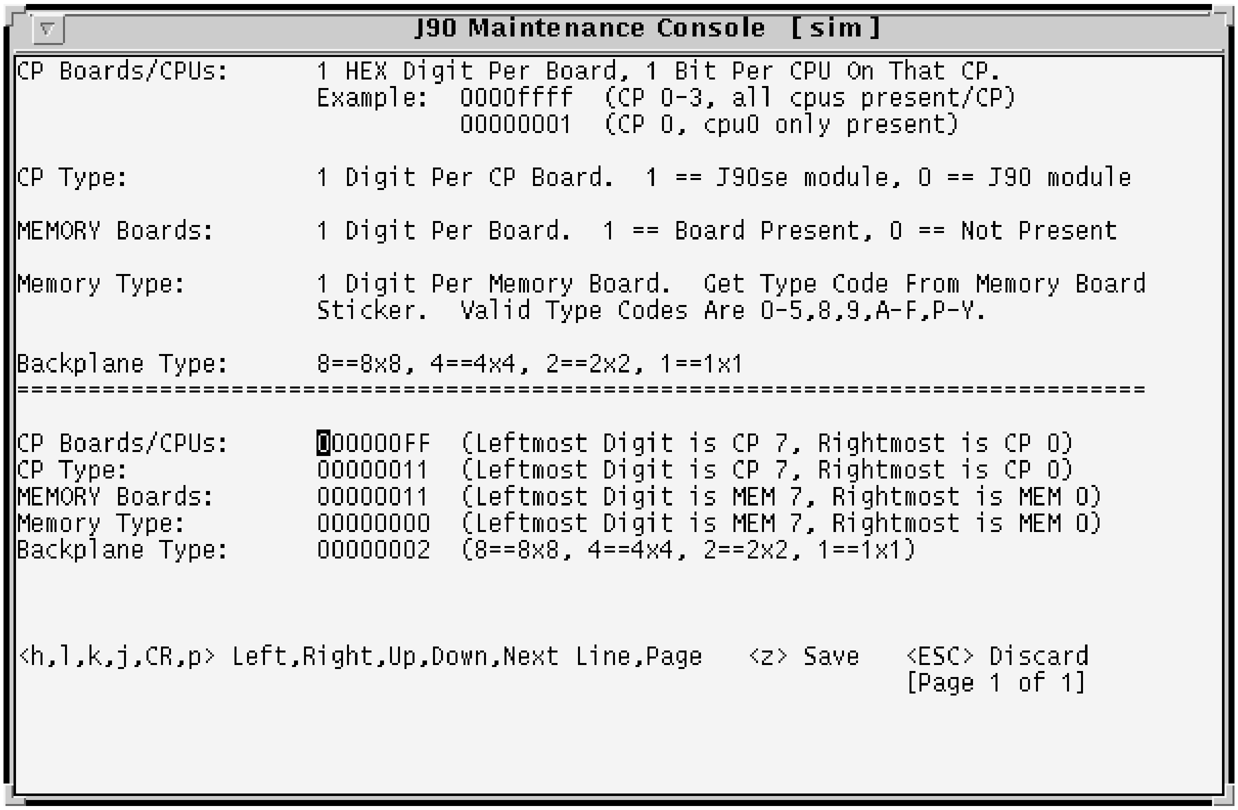
You may use the editor commands to enter information (refer to“Editing jconfig Parameters” on page 66) and enter `z` to save the configuration. At this point, `jconfig` prompts you for verification of the information entered. If no errors are encountered during hardware configuration sensing, `jconfig` prompts you for verification of the hardware configuration and then displays the main menu.

Figure 15. jconfig Hardware Configuration Display



```
***************** System Hardware Configuration ******************

CP BOARDS PRESENT      CPUs PRESENT ON EACH CP BOARD      CP TYPE
-----------------      -----------------------------      -------
0                      0  1  2  3                         J90se
1                      0  1  2  3                         J90se

MEM BOARDS PRESENT:    0  1
MEMORY BOARD TYPES:    0  0

Backplane Type:        2x2
Is This Configuration Correct (y, n, <CTRL-C>)?
```

Figure 16. jconfig Editor Screen



```
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)

CP Type:             1 Digit Per CP Board.  1 == J90se module, 0 == J90 module

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

Memory Type:         1 Digit Per Memory Board.  Get Type Code From Memory Board
                     Sticker.  Valid Type Codes Are 0-5,8,9,A-F,P-Y.

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

CP Boards/CPUs:      000000FF  (Leftmost Digit is CP 7, Rightmost is CP 0)
CP Type:             00000011  (Leftmost Digit is CP 7, Rightmost is CP 0)
MEMORY Boards:       00000011  (Leftmost Digit is MEM 7, Rightmost is MEM 0)
Memory Type:         00000000  (Leftmost Digit is MEM 7, Rightmost is MEM 0)
Backplane Type:      00000002  (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]
```

Editing jconfig Parameters

NOTE: If you are unfamiliar with the `jconfig` utility, you can corrupt the system configuration files and make the system inoperable. This tool should be used only by qualified 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 J90se 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 (refer to Figure 16) 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). After you have made your changes, you must save and update the configuration files.

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. You must save and update the configuration files if you made changes in the editor window.

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

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 8 lists the files that `jconfig` uses.

Table 8. jconfig Files

File	Description
`/opt/config/sn9###/pm[0-7].cfg`	Module-level processor module configuration files that contain ASIC JTAG configuration register fields for that module
`/opt/config/sn9###/mem[0-7].cfg`	Module-level memory module configuration files that contain ASIC JTAG configuration register fields for that module
`/opt/config/sn9###/pm[0-7].cfg.txt`	ASCII version of `pm[0-7].cfg`
`/opt/config/sn9###/mem[0-7].cfg.txt`	ASCII version of `mem[0-7].cfg`

Reference Information

This section provides reference information for the following topics:

- Supervisory Channel
- Maintenance Communication

Supervisory Channel

NOTE: The supervisory channel is a supported maintenance channel in the field only on initial shipments of CRAY J90se series systems. It will not be available for field troubleshooting as the systems become more stable.

The supervisory channel can be used to send GigaRing packets directly to the CRAY J90se client, (bypassing the node interface). The supervisory channel is connected directly to the CRAY J90se system processor module through a special SBus card that plugs into the SWS. The supervisory channel SBus card requires the installation of a device driver. Refer to the SWS file `/opt/CYRIdiag/j90/bin/README.sscc` for more information.

To use the supervisory channel, you must include the `-spvd` option with the maintenance interface command as shown in the following example:

```
act -spvd
```

The actual packets for the supervisory channel are nearly identical to those sent across the GigaRing channel except that they include an additional 64-bit header word in front of the packet to inform the CRAY J90se client where this packet should go. Table 9 shows the possible values for the Supervisory channel header word.

Table 9. Supervisory Channel Header

CRAY J90se Client Destination	Use
0	Write Request
1	Read Response
2	Write Block Done
3	Messages
4	Read Requests, Read/Write Block Init, Read Block Done
5	Read/Write Init Response
6	Maintenance Read/Write (conbus Read/Write)
7	GigaRing Out

Maintenance Communication

The maintenance interfaces (like ACT and JMC) communicate with the CRAY J90se mainframe packets that are sent through an MPN and onto the GigaRing channel. The following paragraphs describe the packets that are used to access the conbus and to initiate DMA transfers.

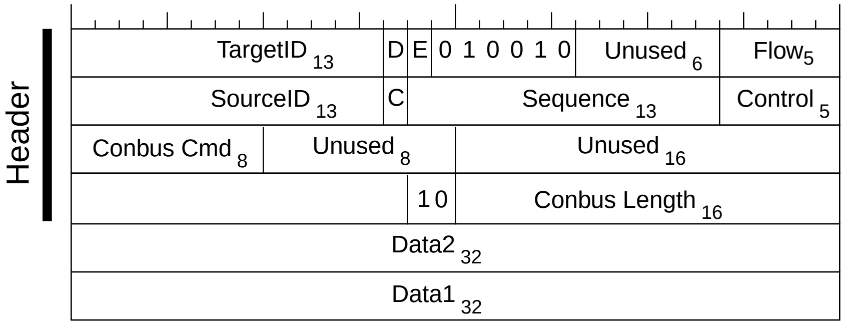
Conbus Packets

There are two types of conbus packets: command packets and data packets. These conbus packets are actually GigaRing maintenance write (MaintWrite) and maintenance read (MaintRead) packets. Conbus packets are used to issue JTAG scan commands for configuring, deadstarting, master clearing, reading shadow registers, and running boundary scan on CRAY J90se series systems.

Conbus Command Packet

The conbus command packet is a MaintWrite request packet with header word 1 bits 17,16 = 10. Header word 1 bits 63 through 56 are the conbus command, payload 0 bits 63 through 32 are Data 2, and payload 0 bits 31 through 0 are Data 1 (refer to Figure 17). A maintenance write response (MaintWrite Resp) packet is returned after the command has been written to the conbus.

Figure 17. Conbus Command Packet



D = Delta (0 = normal addressing, 1 = delta addressing)
C = Corrupt (0 = healthy packet, 1 = corrupt packet)
E = Error (0 = no error, 1 = error)

Conbus Data Packets

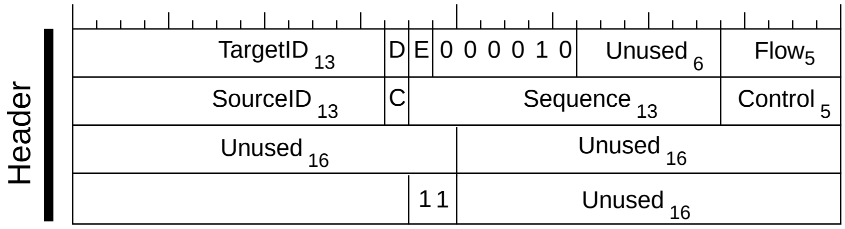
The conbus data is exchanged via GigaRing MaintRead and MaintWrite packets. Figure 18 and Figure 19 show the formats of the conbus write and read data packets.

Conbus Read Data Packet

After a conbus read command packet has been sent, data can be received from the conbus via conbus read data packets. These packets are MaintRead request packets with the header word 1 bits 17,16 = 11. A data payload of 16 words is assumed.

Because the conbus sends data in 128-byte bursts, the data received is payload words 0 through 17 (octal). Bytes are received from word 0 bits 63 through 56, 55 through 48,... to word 17 bits 7 through 0. One MaintRead Response is returned for every MaintRead request that is sent out.

Figure 18. Conbus Read Data Packet



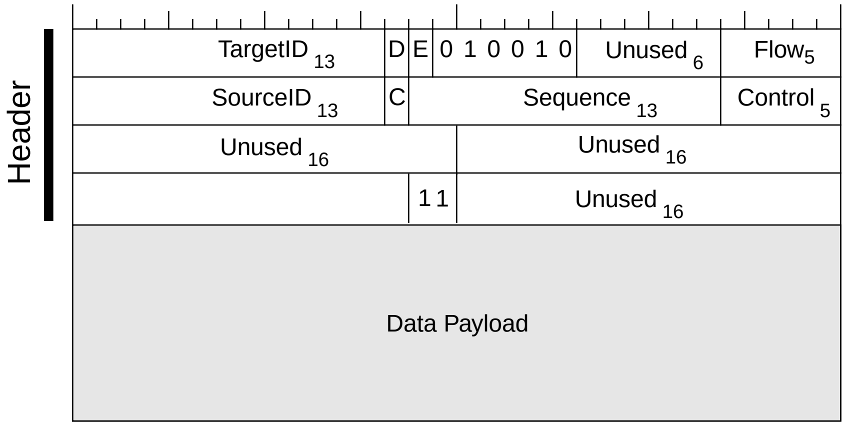
D = Delta (0 = normal addressing, 1 = delta addressing)
C = Corrupt (0 = healthy packet, 1 = corrupt packet)
E = Error (0 = no error, 1 = error)

Conbus Write Data Packet

After a conbus write command packet has been sent, data can be sent to the conbus via conbus write data packets. These packets are MaintWrite request packets with header word 1 bits 17, 16 = 11. A data payload of 16 words is assumed.

Because the conbus receives data in 128-byte bursts, the data sent is in the payload words 0 through 17 (octal). Bytes are sent from word 0 bits 63 through 56, 55 through 48,... to word 17 bits 7 through 0. A MaintWrite response is sent after data has been sent to the conbus. Only 1 conbus write data packet can be outstanding at any time to guarantee that the packets arrive in order. Figure 19 shows the conbus write data packet.

Figure 19. Conbus Write Data Packet



D = Delta (0 = normal addressing, 1 = delta addressing)
C = Corrupt (0 = healthy packet, 1 = corrupt packet)
E = Error (0 = no error, 1 = error)

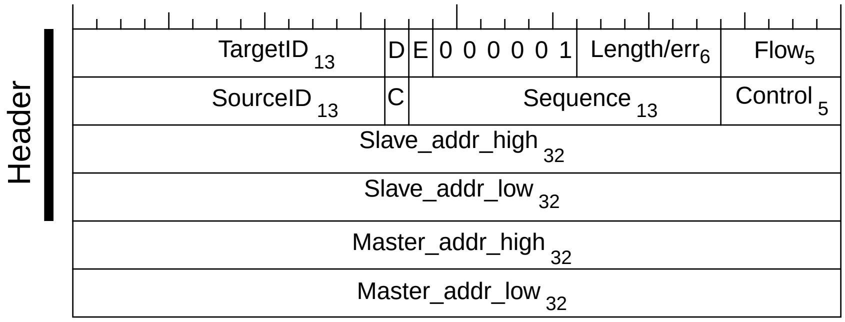
DMA Packets

DMA transfers are performed via standard GigaRing Read and Write packets. The formats and descriptions follow.

DMA Read

The Read packet is sent to the mainframe to request 1 to 32 64-bit words of data from mainframe memory.

Figure 20. DMA Read Packet



<table>
<tr><td>TargetID $_{13}$</td><td>D</td><td>E</td><td>0 0 0 0 0 1</td><td>Length/err$_6$</td><td>Flow$_5$</td></tr>
<tr><td>SourceID $_{13}$</td><td colspan="2">C</td><td colspan="2">Sequence $_{13}$</td><td>Control $_5$</td></tr>
<tr><td colspan="6">Slave_addr_high $_{32}$</td></tr>
<tr><td colspan="6">Slave_addr_low $_{32}$</td></tr>
<tr><td colspan="6">Master_addr_high $_{32}$</td></tr>
<tr><td colspan="6">Master_addr_low $_{32}$</td></tr>
</table>

D = Delta (0 = normal addressing, 1 = delta addressing)
C = Corrupt (0 = healthy packet, 1 = corrupt packet)
E = Error (0 = no error, 1 = error)

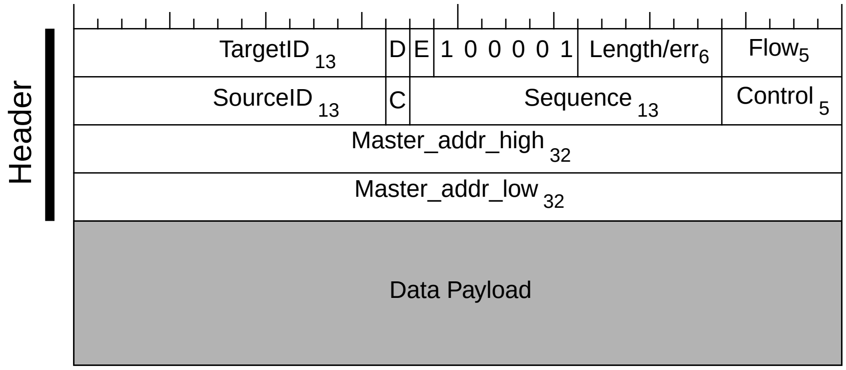
The length field is a 6-bit integer in the range 1 through 32, which represents the number of requested words.

The master address into which the read response data will be placed is passed along with a *Read* request and returned in the ReadResp packet.

ReadResp

The ReadResp packet is sent in response to a *Read*. This packet returns the requested data or an error indication.

Figure 21. DMA Read Response Packet



D = Delta (0 = normal addressing, 1 = delta addressing)
C = Corrupt (0 = healthy packet, 1 = corrupt packet)
E = Error (0 = no error, 1 = error)

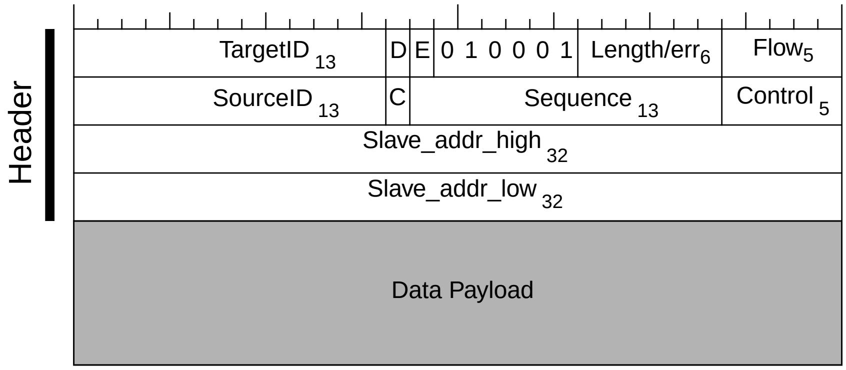
The length field is a 6-bit integer in the range 1 through 32, which represents the number of the returned words.

The master address is used by the originator of the corresponding *Read* to store the returned data.

DMA Write

The Write packet is sent to the CRAY J90se client to store 1 to 32 64-bit words of data into the mainframe’s memory.

Figure 22. DMA Write Packet



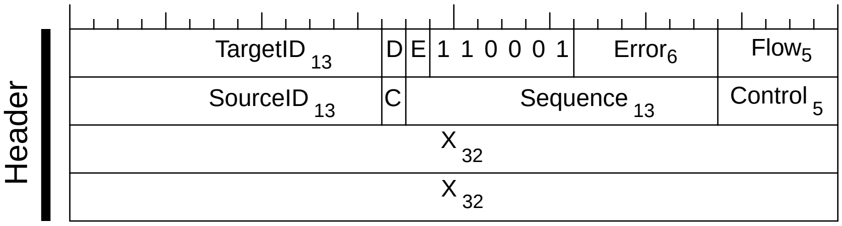
D = Delta (0 = normal addressing, 1 = delta addressing)
C = Corrupt (0 = healthy packet, 1 = corrupt packet)
E = Error (0 = no error, 1 = error)

The length field is a 6-bit integer in the range 1 through 32, which represents the number of payload words.

WriteResp

The WriteResp packet is sent in response to a *Write*. This packet indicates that the *Write* was received by the CRAY J90se client but not necessarily that the data has propagated to the local memory. An error indication can alternately be returned with this packet.

Figure 23. DMA Write Response Packet



D = Delta (0 = normal addressing, 1 = delta addressing)
C = Corrupt (0 = healthy packet, 1 = corrupt packet)
E = Error (0 = no error, 1 = error)
X = Don’t care