CRAY RESEARCH, INC.

CRAY COMPUTER SYSTEMS

I/O SUBSYSTEM (IOS) BASED FIELD ENGINEER'S DIAGNOSTIC REFERENCE MANUAL

HM-1002

RECORD OF REVISION CRAY RESEARCH, INC. PUBLICATION NUMBER HM-1002

Each time this manual is revised and reprinted, all changes issued against the previous version in the form of change packets are incorporated into the new version and the new version is assigned an alphabetic level. Between reprints, changes may be issued against the current version in the form of change packets. Each change packet is assigned a numeric designator, starting with 01 for the first change packet of each revision level.

Every page changed by a reprint or by a change packet has the revision level and change packet number in the lower righthand corner. Changes to part of a page are noted by a change bar along the margin of the page. A change bar in the margin opposite the page number indicates that the entire page is new; a dot in the same place indicates that information has been moved from one page to another, but has not otherwise changed.

Requests for copies of Cray Research, Inc. publications and comments about these publications should be directed to:

CRAY RESEARCH, INC.,
1440 Northland Drive,
Mendota Heights, Minnesota 55120

Revision	Description
	June, 1984 - Original printing.

PREFACE

The maintenance tools described in this manual operate on CRAY-1 Model A, B, C, S, M, and X-MP Computer Systems that use the I/O Subsystem (IOS) to run and monitor diagnostic tests. Although certain models have specific diagnostics associated with them, the Cray maintenance tools themselves operate similarly from one Cray Computer System to the next.

The differences that exist among the maintenance tools themselves are generally restricted to terminology; such as CMOSX and CPXM on Cray X-MP computer systems and CMOS and CPUM on CRAY-1 Computer Systems. For specific information on individual diagnostics, refer to Appendix A in this manual and the Cray Diagnostic Ready Reference Manuals, CRI publications HQ-1007, HQ-1004, and HQ-1005.

HM-1002 is intended for Cray field engineers and is divided into two parts and an appendix section:

Part 1, Operation, describes the commands and capabilities of the CRAY X-MP diagnostic maintenance tools (operating systems, boots, online systems, monitors and disk routines).

Part 2, Software Maintenance, describes the steps that must be taken to assemble or modify diagnostics on site.

The appendix section contains descriptions of CPU, I/O Subsystem (IOS), and foreign interface tests.

Throughout this manual, all keyboard commands are terminated by pressing the RETURN key. See section 4 of the I/O Subsystem (IOS) Operator's Guide, CRI publication SG-0051 for more detailed information about entering commands. All variable or user-supplied command parameters are indicated by the use of italic type. For example, when the following command is entered:

    :SCAN *filename*

The CMOSX :SCAN command is performed on file *filename*.

Other publications that may be of interest to the reader are:

- APML Assembler Reference Manual, CRI publication SR-0036
- APML Quick Reference Card, CRI publication SQ-0059
- CAL Assembler Version 1 Reference Manual, CRI publication SR-0000

- COS Operational Aids Reference Manual, publication SM-0044
- CRAY-1 Computer Systems Diagnsotic Ready Reference Guide, publication HQ-1004
- CRAY-1 Computer Systems Maintenance Control Unit (MCU) Based Field Engineer's Diagnostic Reference Manual, HM-1000
- CRAY-1 S Series Mainframe Reference Manual, publication HR-0029
- CRAY-1 M Series Mainframe Reference Manual, publication HR-0064
- CRAY-OS Version 1 Reference Manual, publication SR-0011
- CRAY X-MP Computer Systems Diagnostic Ready Reference Guide, publication HQ-1005
- CRAY X-MP Series Mainframe Reference Manual, publication HR-0032
- I/O Subsystem (IOS) Diagnostic Ready Reference Guide, publication HQ-1003
- I/O Subsystem (IOS) Operator's Guide, publication SG-0051
- I/O Subsystem Reference Manual, publication HR-0030
- IOS Software Internal Reference Manual, publication SM-0046
- Diagnostic Programmer's Guide, publication CP-1006
- UPDATE Reference Manual, publication SR-0013

CONTENTS

FIGURES

TABLES

PART 2 - SOFTWARE MAINTENANCE

FIGURES

APPENDIX

FIGURES

TABLES

PART 1

OPERATION

INTRODUCTION 1

Cray Research, Inc., diagnostics use the maintenance control unit (MCU) philosophy. The diagnostics use a small computer or I/O Subsystem (IOS) to test a CPU or another I/O Processor (IOP) (see figure 1-1).

The program modules associated with display, control, and problem isolation are executed in the MCU, and not in the suspected failing machine. The family of diagnostic tools for the CRAY X-MP and CRAY-1 S and M Series of Computer Systems is based in the I/O Subsystem (IOS) and made up of the following programs:

- Diagnostic Support System (DSS)
- Cray Maintenance Operating System (CMOSX)
- CPU and I/O Subsystem (IOS) boots (various stand-alone tests)
- CPU and IOS online tests
- Disk aid routines

The diagnostic software described in this manual runs on Cray Computer Systems and requires the following minimum hardware configuration:

- IOP0 - Master I/O Processor (MIOP)
- 2 CRT-4 displays
- Peripheral Expander with a tape drive†
- 80 Mbyte disk
- Printer††

Detailed information on the IOS is beyond the scope of this publication. See the I/O Subsystem Reference Manual, CRI publication HR-0030, the I/O Subsystem (IOS) Operator's Guide, CRI publication SG-0051, and the IOS Software Internal Reference Manual, CRI publication SM-0046 for additional information.

† Only required when installing a new system or a bugfix
†† Useful but not required for most diagnostics. Online software has different requirements.

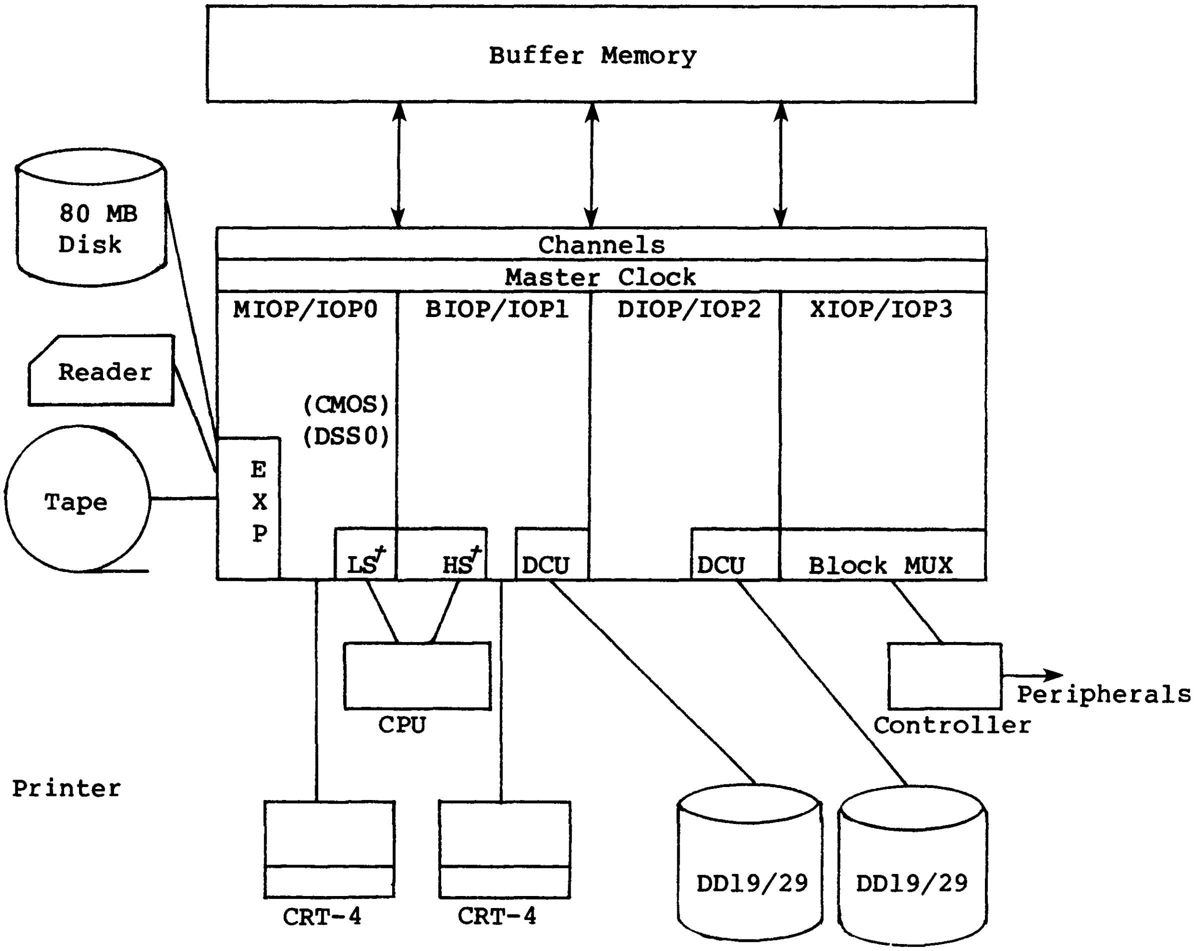


Figure 1-1. I/O Subsystem configuration

1.1 DIAGNOSTIC SUPPORT SYSTEM (DSS)

The Diagnostic Support System (DSS) is a simple operating system that performs file management for diagnostic tests. DSS performs transfers between modules (CMOSX and boots), loads diagnostic binaries, dumps backup copies of diagnostic binaries, and edits command buffers. DSS executes in IOP0 or IOP1 and, therefore, can also be referred to as DSS0 or DSS1. For a detailed description of DSS, see part 1, section 2.

† HS and LS represent the high-speed and low-speed channels, respectively.

1.2 CRAY MAINTENANCE OPERATING SYSTEM

The Cray Maintenance Operating System (CMOSX for CRAY X-MP Computer Systems or CMOS for CRAY-1 S and M Computer Systems) is an MCU system control program, based in I/O Processor 0 (IOP0), that can run diagnostics concurrently on CPUs and IOPs. CMOSX and its tests are loaded from the Peripheral Expander 80 Mbyte disk, tape drive, or the micro MCU into IOP0.

If CMOSX cannot be brought up, use the basic checkout display (BCD) program or the basic processor diagnostic (BPX). BCD and BPX exist on stand-alone deadstart tapes See part 1, section 6 of this manual for a detailed description of these two programs.

A micro MCU-based version of MCU Basic can also be used to test IOP0.[†] See the Cray-1 Computer Systems Maintenance Control Unit (MCU) Based Field Engineer's Diagnostic Reference Manual, publication HM-1000 for a detailed description of MCU Basic. For a detailed description of CMOSX, see part 1, section 3 of this manual.

1.3 CPU AND I/O SUBSYSTEM BOOTS

CPU and I/O Subsystem boots are individual stand-alone tests that do not run under the control of MCU systems. They are, instead, bootstrapped into an IOP and take over the display. Examples of boots are CPXM or CPUM (Cray CPU Memory test), IOPM (I/O Processor Memory test), and BMT (Buffer Memory test). CPU boots are detailed in part 1, section 4. For a detailed description of IOS boots, see part 1, section 6

1.4 CPU AND I/O SUBSYSTEM (IOS) ONLINE TESTS

Online diagnostics tests run under the control of the customer's operating system; normally the Cray Operating System (COS). To submit an online diagnostic test, enter a series of COS control statements as a COS job. For more information about COS control statements, see the CRAY-OS Version 1 Reference Manual, publication SR-0011. Online CPU tests are detailed in section 5. For a detailed description of online IOS tests, see part 1, section 8.

† A NOVA-4C based version of MCU Basic can also be used to test IOP0.

1.5 DISK AID ROUTINE

The disk aid routine is a micro-interpreter that allows you to interact with disk drives and the controller without being concerned with the detailed programming of an I/O sequence. For a detailed description of disk aid routines, see part 1, section 7.

DIAGNOSTIC SUPPORT SYSTEM (DSS) 2

The Diagnostic Support System (DSS) is a simple operating system that provides the following file management functions for all CMOSX tests:

- Organizes files (File Name Table)
- Manages file-to-file functions (file commands)
- Loads diagnostic program modules
- Edits diagnostic program text
- Manages the tape file directory

DSS runs in IOP0 and accesses the Peripheral Expander 80 Mbyte disk drive. DSS is called DSS0 when it is running in IOP0. DSS0 can be intialized from either tape or disk and is used to run diagnostics tests on IOP1, IOP2, and IOP3. If deadstarting from tape, use the tape deadstarting procedures listed in Appendix D.1 and select the following from the boot menu:

DSS0

If deadstarting from disk, follow the disk deadstart procedures in Appendix D.2. The disk deadstart procedure automatically brings up DSS0.

DSS1† can also be brought up from either tape or disk and is used to test IOP0, the line printer, and the Peripheral Expander chassis. If deadstarting from tape, follow the tape deadstart procedures in Appendix D.3.1 and select the following from the boot menu:

DSS1

Identify the IOP1 console and the disk by entering U on an IOP1 console. The screen then prompts you to designate a DD-19/29 disk channel by displaying the following query:

CHANNEL?

---

† DSS1 is only available on Cray Computer Systems that are configured with DD-19s and DD-29s. Sites not able to run DSS1 can use the micro MCU to test IOP0.

Respond to the query by entering a value in the range between 20_8 and 37_8.

If deadstarting from disk, follow the disk deadstart procedures in Appendix D.2. The disk deadstart procedure automatically brings up DSS0. To boot DSS1, enter:

BOOT DSS1.

Identify the IOP1 console by entering U on the IOP1 console. The screen then prompts you to designate a DD-19/29 disk channel by displaying the following query:

CHANNEL?

Respond to the query by entering a value in the range between 20_8 and 37_8.

NOTE

Running DSS1 requires much of IOP0 to be functional.

2.1 FILE ORGANIZATION

DSS organizes program files by tracks on disk in the following way:

- Track 2 File Name Table (FNT) 2 contains the program modules associated with DSS.
- Track 3 FNT 3 contains the I/O Subsystem (IOS) tests.
- Track 4 FNT 4 contains all of the CPU tests.
- Track 5 FNT 5 contains the text files used by the HELP commands.

2.2 DSS FILE MANAGEMENT COMMANDS

DSS manages individual CMOSX files with DSS file management commands. DSS returns a running dayfile of the file management commands that have been entered. No other displays are associated with DSS. To obtain more information on the other features available with DSS, enter the following command:

LIST DSS 5

Press the RETURN key to terminate all DSS commands. Use the DEL key to backspace over typographical errors, or use the LINEFEED key to erase the entire line. Table 2-1 lists DSS file management commands and briefly describes each.

Table 2-1. DSS file management commands

Command	Description
ADD *filea fileb*	Concatenates *filea* and *fileb*; contents of *filea* are unchanged.
APAL *infile outfile listfile*	Invokes the APAL assembler; the APML source statements are stored in *infile*, the output in *outfile*, and the listing is created in *listfile*.
APPEND *testname monitor dfile nblks*	Unblocks diagnostic *testname* and monitor *monitor*, then writes the output to disk file *dfile*. *nblks* is the number of 4000-word blocks to write; the default is 1.
BITS *binfile listfile*	Creates a listing file from a binary disk file. The contents of the listing file are formatted into three different bit patterns as shown in the example below: 1 000 111 000 111 000 (1-3-3-3-3-3 pattern) 10001110 00111000 (8-8 pattern) 1000111000111000 (1 by 16 pattern) *binfile* is the name of the binary disk file and *listfile* the listing file that is created. The primary purpose of this program is to display program code in a format that allows efficient decoding.
BOOT *name*	Brings up the stand-alone module *name* where *name* can be the name of any module that is intended to be booted. For example, any of the following:

Table 2-1. DSS file management commands (continued)

Command	Description
BOOT (continued)	BMT† CBM CLRIO CMOS CMOSX CPXM0 DKX DUMP IOPM† IPC† MBUF† SLAT
BTOL *infile binfile n*	Creates octal listing *binfile* from binary input file *infile*. *n* is the FNT track number; the default is 2. Each binary word in the source file is formatted into a 6-digit grouping that is printed. The binary bits are grouped in a 1-3-3-3-3-3 pattern for printing. If *infile* is a number, the corresponding track is used as binary input.
CDC *num*	Changes the display image to the image on channel *num*
CDFL	Clears all but the last track of the dayfile
CFNT *n*	Drops all files in FNT track *n*
COPY *oldfile newfile* *oldfnt newfnt*	Copies all data blocks of *oldfile* on disk to *newfile*. If filename *newfile* already exists on disk, it is purged. *oldfnt* is the the FNT track number of *oldfile* and *newfnt* is the FNT track number of *newfile*. If *oldfnt* and *newfnt* are not specified, the default is 2.
COPYT	Duplicates a tape (IOP0 deadstart tape or DSS initialization tape). Use the following procedure: 1. Mount the source tape. 2. Press the SPACEBAR to read it. 3. The tape rewinds when it has been read. 4. Mount the destination tape. 5. Press the SPACEBAR to write it. 6. Press the RETURN key to terminate the current write operation.

† See part 1, section 6 for a detailed description of this diagnostic.

Table 2-1. DSS file management commands (continued)

Command	Description
COPYT (continued)	If any read or write errors occur during the copying process, the program pauses and displays an appropriate message on the screen. The program can be aborted at this point by pressing the RETURN key. Program execution continues with a retry if the SPACEBAR is pressed. Files that have been replaced and labeled as obselete (subsection 2.5) in the tape file directory are not copied, unless the pound sign is the first input parameter.
CVRT *dgfile dssfile n*	Converts DG text format *dgfile* to DSS text format *dssfile*. *n* is the FNT track number; the default is 2.
CVRTA *dssfile dgfile n*	Converts DSS text format *dssfile* to DG text format *dgfile*. *n* is the FNT track number; the default is 2.
DDF	Displays the last 20 lines of the dayfile on the screen
DISK	Displays the number of disk sectors used on the screen
DROP *filename n*	Removes *filename* from the FNT. All disk tracks occupied by the file are freed for use by other files. The file index block (FIB) is removed; all file contents are lost. Once the DROP utility has been used on a file, the file cannot be restored. *n* is the FNT number; the default is 2.
DSA *filename*	Deadstarts another I/O Processor (IOP) with file *filename*†.
DSA ECD	Deadstarts the Peripheral Expander driver to IOP0. DSA ECD must be entered prior to using tape, a reader, or a printer.

† DSS0 deadstarts IOP1; DSS1 deadstarts IOP0.

Table 2-1. DSS file management commands (continued)

Command	Description
DSA LOG	Deadstarts the error log display program into IOP0. Enter U to identify the IOP0 display console.
DSA TBOOT	Redeadstarts IOP0 from tape
EDIT *oldname newname*	Allows the user to examine and modify the contents of a disk storage file on the screen. If this program is used, it is unnecessary to keep backup files on punched cards, because tapes can be used to store backup files. *oldname* is the name of the file to be edited, and *newname* is the name of the file that is created and edited. @ can be used in place of the input file *oldname*. EDIT @ creates file *newname* without any input file.
ERASE	Erases an entire tape
ETYPE *file*	Displays error lines from the A Programming Assembly Language (APAL) listing *file* on the display. To advance to the next error line, press the SPACEBAR. To abort, press the CR key.
FDMP *n*	Dumps all program modules from one FNT to a backup tape where *n* can be one of the following: *n*=2 Dumps DSS program and boots *n*=3 Dumps I/O Subsystem program *n*=4 Dumps CPU programs *n*=5 Dumps the text FNT
FLOAD *n*	Loads an FDMP-type tape into FNT *n*. FLOAD is the only program that can read the tape that is created.
FLOAD @	Loads all files into their proper FNTs as specified by the tape's FNT directory

Table 2-1. DSS file management commands (continued)

Command	Description
FNT *n dfile*	Formats the FNT into a new disk file that can be listed or edited. *n* is the FNT track number; default is 2. If the file already exists on the disk, it is replaced by the new file. If the destination file *dfile* is not entered, the formatted file is displayed on the screen.
LIST *listfile n*	Prints a listing from file *listfile* on FNT *n*. Binary files can be listed in octal by executing the DSS BTOL command and then listing the resulting file. *n* is the FNT track number; default is 2. The program pauses at the end of a page if any key is pressed and can be aborted at this point by pressing the CR key. Program execution continues if the SPACEBAR is pressed. @ can be used as the file name. If @ is used, the dayfile, which is not accessible by name, is listed.
LISTE *filename*	Lists error lines from the APML listing file *filename*
LLA *n*	Loads all files from an Eclipse MCU Basic library tape to FNT track *n*; default is 3.
LLF *testname n*	Loads program *testname* to FNT track *n* from an Eclipse MCU Basic library tape
LOAD *filename*	Copies a card deck to disk storage and labels the file *filename*. Data is packed as it is loaded and an end of file is marked by an ETX that is sensed in the card data.
MLOAD	Loads the Nova portion of the disk from an MCU Basic save tape
READ *sfile newfile n*	Reads a source file *sfile* from tape and stores the file on disk. If the file is not found, a message is printed on the screen

Table 2-1. DSS file management commands (continued)

Command	Description
READ (continued)	and the program terminates. If *sfile* is found in the directory, the file is located on tape. A request/write operation is performed to establish the new disk file in the FNT and create a file index block (FIB) for the file. *newfile* is the name of the new file on disk. Data is written to disk in blocks, and the tape is rewound after all data has been transferred. The Track Reservation Table and FIB are stored for the file. *n* is the FNT track number; the default is 2. @ can be substituted for the name of the file on tape. When this is done, the program assumes that the tape is already properly positioned at the point where reading is to start.
READA *filename n*	Reads all files from tape to FNT track *n*; the default is 2. If *filename* is a character name, all files are concatenated to it.
RENAME *old new n*	Allows the names of disk files to be changed. *old* is the previous name of the disk file, *new* is the label that is to be associated with the file, and *n* is the FNT track number; the default is 2. If *new* already exists in the FNT, the request is not honored and an error message is displayed.
REWIND	Rewinds the Peripheral Expander tape drive
RTDIR *filename*	Reads a tape directory to disk file *filename*. Once on disk, the directory can be edited and rewritten to the tape.
RTL	Reads and writes data files using the tape directory. RTL is part of the deadstart package and has no input parameters. Data files are written to disk; files replaced

Table 2-1. DSS file management commands (continued)

Command	Description
RTL (continued)	and labeled as obsolete (subsection 2.5) are skipped. The name of each file is entered in the FNT, and a file index block (FIB) is created for each data file.
RXT *name fileno n*	Copies RDOS XFER tape file *fileno* to file *name* on FNT track *n*; the default is 4.
SKIPF	Skips one tape file
SORT *n*	Sorts the file names in FNT track *n*
TDMP	Creates an IOP0 deadstart tape that stores the exit stack, operand registers, or Local Memory. Boot the tape, and edit it using the commands listed below: TE — Types the contents of the exit stack TO *m n* — Types the contents of operand registers *m* through *n* TM *a b* — Types the contents of Local Memory addresses *a* through *b* PE — Prints the contents of the exit stack PO *m n* — Prints the contents of operands *m* through *n* PM *m n* — Prints the contents of Local Memory addresses *m* through *n* The program uses memory addresses 0 through 32, 76000 through 100524, and operand register 776.
TYPE *filename n*	Displays the contents of file *filename* from FNT track *n* on the screen; the default for *n* is 2. The program pauses for a form feed character or keyboard interrupt. To continue, press the SPACEBAR; to abort, press the CR key.
VERIFY *filea fileb*	Compares the data of disk files *filea* and *fileb*. An error condition occurs when the

Table 2-1. DSS file management commands (continued)

Command	Description
VERIFY (continued)	data in one file does not agree with the data in the second file, or when the two files are not of the same length. When an error condition is sensed, the program terminates with an error message.
WDSP *tapename progname library dfile*	Writes tape deadstart package *tapename* to disk file *dfile*. *progname* (disk deadstart program) and *library* (tape library read program) are the programs to be written.
WDSR *recname dfile m n*	Writes deadstart record *recname* of *m* 4000-word blocks to disk file *dfile* in FNT track *n*. The default for *m* is 1 and for *n* is 2.
WEOF	Writes a file mark on tape
WLA *n*	Dumps all files in FNT *n* to an Nova Eclipse MCU Basic library tape
WRDIAG *diagname monitor tape*	Unblocks diagnostic *diagname* and monitor *monitor* and writes the data to a 10000-word tape record. If *tape* is included, a second 10000-word tape record is written.
WRITE *diskname tapename n*	Writes disk file *diskname* to tape file *tapename*. FNT searches for *diskname*. If *diskname* is not found, an error message is displayed and the program is terminated. A tape directory must exist before the WRITE command can be executed (the directory may be blank). The tape directory is searched to make sure that *tapename* does not already exist on tape. If it does, the existing entry is replaced and labeled as obsolete (subsection 2.5). After all data is transferred, the tape is rewound. *n* is the FNT track number to be searched; the default is 2.

Table 2-1. DSS file management commands

Command	Description
WRITEA *filename*	Writes all files in file *filename* to tape. *filename* must have the following format: *diskname tapename n -comments* *diskname* is the file on disk and *tapename* is the file on tape. If *tapename* is not specified, the tape file is given the name of the disk file. *n* is the name of the FNT track number; the default is 2. *comments* are optional and must be preceded by a dash if they are included.
WTDIR *diskname*	Writes a tape directory from disk file *diskname*. A tape directory is usually maintained by writing files to tape, occasion can arise where editing the directory is desired. This program replaces the directory on tape. A blank tape directory may be written on tape by using EDIT@ to crate a file with an appropriate heading but no file names (Tape Directory, Date, and so on).

2.3 DIAGNOSTIC MODULE LOADER

Stand-alone program modules can be loaded with the following DSS file management commands:

- DSA *filename* — Deadstarts file *filename* into another IOP. If IOP0 is running, IOP1 is deadstarted. If IOP1 is running, IOP0 is deadstarted. For more examples of the DSA command, see DSS file management commands in this section.
- BOOT *filename* — Replaces the current copy of DSS with file *filename*. BOOT then executes *filename*.

2.3.1 LOADING MODULES WITH THE DEADSTART TAPE

The following options are available from the IOP0 deadstart tape. The program name followed by a RETURN begins execution.

Test	Description
BMT	Buffer Memory test
CBM	Clear Buffer Memory
CLRIO	Clear IOP
CMOSX	Cray Maintenance Operating System
CPXM0	CPU memory test (6 Mbyte channel)
DKX	Disk diagnostic test (DD-19 and DD-29)
DSS0	Diagnostic Support System (IOP0 based)
DSS1	Diagnostic Support System (IOP1 based)
DUMP	Dump to printer
IOPM	I/O Processor memory test
IOPMA	I/O Processor memory boot (2AS module)
IPC	Inter-processor channel test
MBUF	Multiprocessor Buffer Memory test
SLAT	Synchronous line adaptor test

2.3.2 LOADING BOOTS FROM DSS0

Programs can be called from DSS0 by entering the following command:

BOOT *name*

name can be any bootable file on FNT 2 or one of the following:

Test	Description
BMT	Buffer Memory test
CLRIO	Clear IOP
CMOSX	Cray Maintenance Operating System
CPXM0	CPU memory test (6 Mbyte channel)
DKX	Disk diagnostic test (DD-19 and DD-29)
DUMP	Dump to printer
IOPM	I/O Processor Memory test
IOPMA	I/O Processor Memory test
IPC	Inter-processor channel test
MBUF	Multiprocessor Buffer Memory test
SLAT	Synchronous line adaptor test

2.4 DSS TEXT EDITOR

The DSS text editor provides a convenient way to manipulate text files such as command buffers. Use the DSS text editor to modify an existing file by entering:

EDIT *name newname*

See table 2-1 for a description of the *name* and *newname* parameters for the EDIT command. Create a new file with the following command:

EDIT @ *newname*

The DSS text editor works on files located in FNT 2. Command buffers must be copied to FNT 2 before using the EDIT command and then copied back to the original FNT after editing (see the COPY command). The keys listed in table 2-2 control edit functions.

Table 2-2. Edit function keys

Key	Function
DEL	Clears the current character
ESC	Advances the cursor to the next tab position
LF (LINEFEED)	Advances the display one line
CR (RETURN)	Advances the display one line and inserts a blank line
RUB OUT	Clears the current character
SPACE	Advances the cursor to the next character position
CTRL-A	Begins recording mode at the top of the screen
CTRL-B	Ends recording mode at the top of the screen
CTRL-C	Clears the edit line
CTRL-D	Deletes the edit line
CTRL-E	Advances the display 24 lines
CTRL-F	Advances the display to the top of the next page
CTRL-G	Resets the display to the top of the screen
CTRL-H	Backspaces one character
CTRL-I	Spaces forward until the SPACEBAR is depressed
CTRL-K	Spaces forward to the end of text and ends the editing session
CTRL-L	Scans forward at high speed until the SPACEBAR is depressed
CTRL-N	Sets a tab at the cursor position
CTRL-O	Clears the tab at the cursor position
CTRL-P	Backs up the display one line
CTRL-Q	Inserts a blank character at the cursor position
CTRL-R	Deletes a character at the cursor position

Table 2-2. Edit function keys (continued)

Key	Function
CTRL-S	Searches for the string that is entered after CTRL-S is pressed
CTRL-T	Changes the current character to lowercase
CTRL-Y	Merges files

2.5 TAPE FILE DIRECTORY MANGAGEMENT

DSS manages the tape file directory; the first data block following the deadstart package on a magnetic tape. If the deadstart package is not on the tape, the tape file directory is the first data block on the tape.

The directory is composed of a series of entries. One entry exists in the directory for each data file on the tape. The first six characters of each entry contain the name of the data file. The rest of the entry is devoted to comments.

Data files are stored on tape, after the tape file directory, in the order that the file names appear in the directory. When a new file is written to a tape, the new entry is placed in the next available position in the directory. The file name is stored in the first three words of the new directory entry, and the new data file is written in the next available location on the tape.

If a file is written to a tape and a data file of the same name exists on the tape, the previous directory entry is labeled with an @ in the first 8 bits of the directory entry for that file. The new entry is placed in the next available location in the directory, and the data file is written in the next available location on the tape. The old data file is not purged, but is disregarded because of the presence of the @ in the first character position of the directory entry.

The tape file directory can be edited, like any other data file, using the DSS EDIT command (DSS file management commands). To edit the program, read the file to disk using the DSS RTDIR command (DSS file management commands). When the EDIT command is invoked, the disk file name is given to the directory as the source file name.

Files can be disabled by entering an @ in the first position of the file name. Disabled files can be enabled by removing the @ from the first position of the file name and renaming the file to avoid conflicts with existing file names. The edited tape directory is written back to the tape using the DSS WTDIR command (DSS file management commands).

CRAY MAINTENANCE OPERATING SYSTEM (CMOSX) 3

3.1 INTRODUCTION

The Cray Maintenance Operating System (CMOSX† for CRAY X-MP Computer Systems and CMOS for CRAY-1 S and M Series Computer Systems) is a diagnostic operating system that allows you to run diagnostics on Cray Computer Systems. CMOSX contains programs to display and control diagnostic tests as well as utilities and aids for troubleshooting.

CMOSX runs in the Master I/O Processor (MIOP or IOP0) and uses either the AMPEX disk or the Peripheral Expander chassis tape drive as a library device to store files. CMOSX uses the 6 Mbyte channel to the CPU from IOP0 to run and monitor CPU diagnostics. A typical I/O Subsystem (IOS) hardware configuration is illustrated in figure 3-1.

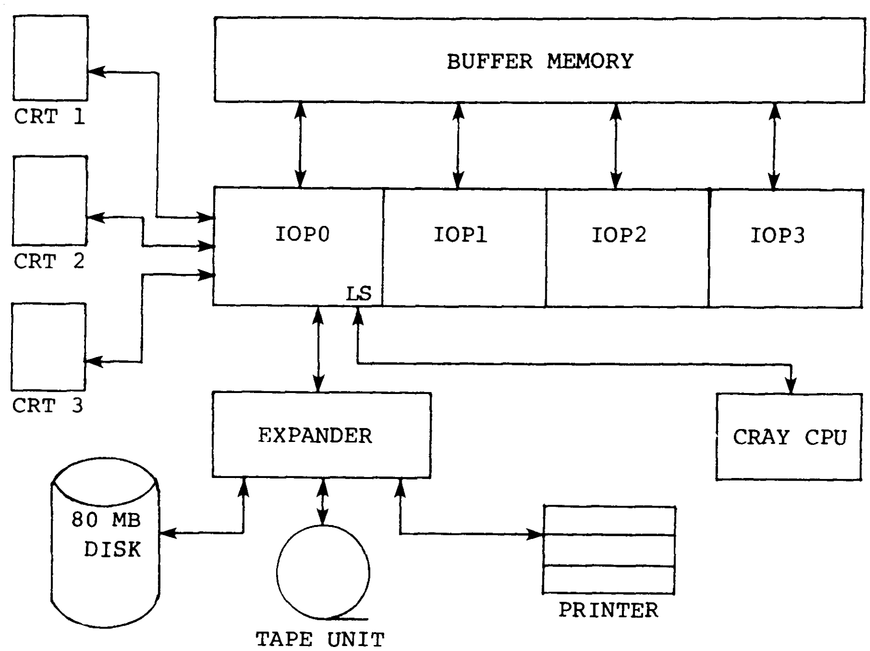


Figure 3-1. Typical hardware configuration

† This manual always refers to the Cray Maintenance Operating System as CMOSX. All references to CMOSX are applicable to CMOS. Any differences between CMOSX and CMOS are specifically noted in the text.

CMOSX is actually three separate subsystems that share the same device drivers, calls, and utilities.

- Subsystem 0 IOS maintenance system
- Subsystem 1 Cray CPU maintenance system
- Subsystem 2 Error logging system (available on CRAY-1 Computer Systems through serial 24 only)

CMOSX coordinates the activities of the three subsystems by providing the following:

- Control over subsystem initialization
- A common filing system
- Consistency through similarities among the subsystems
- A set of display and execution control commands
- Internal Configuration Tables

3.1.1 SUBSYSTEM INITIALIZATION

CMOSX controls the three subsystems and must be operating before any of the individual subsystems can be selected. To deadstart CMOSX into IOP0 under DSS, enter:

```
BOOT CMOSX
```

The IOS maintenance subsystem appears on the console when CMOSX is booted in. The IOS maintenance subystem is Subsystem 0, the Master Subsystem. The other subsystems can be brought into execution using the subsystem control commands from the Master Subsystem.

CMOSX uses from one to three of the consoles attached to IOP0. Each of the subsystems can display output on a separate console and can act as an independent system. System independence allows diagnostics to run simultaneously on IOP1, IOP2, IOP3, and the Cray CPUs. The subsystems share and compete for use of the Peripheral Expander and its peripherals.

A tape version of CMOSX (CMOSX/T) is also available, but is intended as a backup to the disk system and does not contain all of the features available on the disk system. The files on CMOSX/T are arranged so that IOS diagnostics are accessed before Cray CPU diagnostics.

NOTE

Loading diagnostics using CMOSX/T can be a time consuming activity if the desired file is near the end of the tape, and should only be considered as a last resort.

3.1.2 COMMON FILING SYSTEM - FILE NAME TABLE (FNT)

Individual diagnostic programs and support files used by CMOSX are stored on disk and grouped in directories called File Name Tables (FNTs). Each FNT can contain up to 512 files. Diagnostic programs and files have been assigned to the following FNTs:

FNT	Significance
2	File Name Table (FNT) 2 contains the program modules associated with DSS.
3	IOS diagnostics (see Appendix B) and files called from the Subsystem 0 console
4	Cray CPU diagnostics (see Appendix A) and files called from the Subsystem 1 console
5	Documentation and files called from the HELP utility

3.1.3 SIMILARITIES AMONG THE SUBSYSTEMS

This subsection describes the similarities that provide consistency among the subsystems in CMOSX. Those similarities include:

- Contol commands
- Memory displays

Subsystem CONTROL commands

The following commands control the subsystems within CMOSX and can be entered at any console. All console channel numbers are optional. If the channel number is omitted, CMOSX waits for a U to be typed on the selected console. Table 3-1 lists CMOSX subsystem CONTROL commands and briefly describes the function of each.

Table 3-1. CMOSX subsystem CONTROL commands

Command	Description
:BO *file*	Terminates CMOSX, loads, and executes the elected *file* in IOPO. The default file is the disk or tape bootstrap loader.
:BY	Ends the subsystem displayed
:CR *ch*	Invokes the Cray subsystem on console channel *ch*
:ER *ch*	Invokes the error log subsystem on console channel *ch*†
:IO *ch*	Invokes the IOS subsystem on console channel *ch*††

† The error log channel is only available on the CRAY-1 Models A, B, C and the CRAY-1 S S/N 20 and below.

†† Subsystem 0 is normally active but need not be. If it is necessary to run from only one console, Subsystem 0 can be terminated in favor of Subsystems 1 or 2. Generally, you can end a subsystem in favor of another without affecting the diagnostics currently running.

Memory displays

CMOSX memory displays are divided into a left side and a right side. To specify the first word address to be displayed on each side of the screen, enter a left address and/or right address with any subsystem display command. Each side can be further divided to form quadrants (upper left, lower left, upper right, and lower right) that can display different addresses, memory types, and/or memory formats. For example, to display a specific address in a quadrant, enter an upper address and/or lower address with the command as shown below:

DR *upperaddr loweraddr*

The command shown above sets the first word addresses to be displayed on the console in the upper and lower right quadrants. For a complete list of display commands, see subsection 3.1.4.

Type of memory is the kind of memory being displayed. Memory types are: Local, Buffer, CPU, and MCU. The letter designators above the data on either side of the screen give the memory types for that side of the screen. The top letter designator is used for the upper quadrant and the bottom for the lower quadrant, (see figure 3-2). Memory type letter designators are as follows:

Code	Memory type
L	Local
B	Buffer
C	CPU
M	MCU

```
 MODE TM   PN 1               CMOSX/I-I  1.0       09/27/82      :ADB
                                                   11:10:45    > :ADB
 L                                      L
 L                                      L
 0   070043 000000 000000 000000       200  010000 154002 010000 024014
 4   000000 000000 000000 000000       204  010001 024015 014000 065432
10   000001 000400 000000 000000       210  024023 014000 043762 024021
14   000000 000000 000000 000000       214  014000 072270 024022 020014
20   000000 000000 000000 000000       220  024010 020015 024011 072112
24   000000 000000 000001 000000       224  020014 022015 024013 072205
30   000000 000000 000000 000000       230  103070 020015 022014 024013
34   000000 000000 000000 000000       234  020016 024012 072176 103061

40   001000 000000 000000 150002       240  020015 005001 024015 104024
44   024776 010000 154002 014000       244  010001 024015 020014 005001
50   000124 155002 020776 154002       250  024014 024010 101011 010001
54   010027 024776 030776 103017       254  024011 072060 010001 024015
60   010001 154002 010200 155002       260  020012 024014 071043 010024
64   010031 024776 010035 024777       264  024020 036020 076777 072017
70   030777 034776 027776 027777       270  020021 024010 024014 072013
74   030777 034776 040005 104001       274  020021 024011 024015 072036

 ENTER  ?
```

Figure 3-2. Subsystem 0 display

Format is how data is displayed. The format is usually set to display the actual contents of memory addresses. However, memory data can be formatted into more meaningful information. Three memory formats are available:

- Octal memory
- Text
- Exchange Package

Octal memory can be displayed in parcel mode or in word mode. The format designator (a blank space for Actual Memory, X for Exchange Package, or T for text) appears next to the memory type designator and is followed by an address. Actual memory is the default memory type.

3.1.4 CMOSX COMMANDS

Display commands are available for both CPU and IOS displays. Display commands unique to a particular subsystem are listed in the subsection devoted to a particular subsystem. CMOSX commands can be entered from any of the subsystems and in any order desired. The following CMOSX commands are used to control displays, manipulate files, execute tests, and print output.

- FORMAT
- TEXT
- ROLL
- MODE
- REFRESH control
- DATE and TIME
- BASE ADDRESS
- FILE manipulation
- Miscellaneous

CMOSX display FORMAT commands

All display commands are subsets of the basic display FORMAT command as shown in table 3-2. The term subset is not intended to imply an entry sequence or a hierarchy among the display commands. In this case, the term subset represents a consistency among the commands in the way they are constructed and entered.

Table 3-2. CMOSX display FORMAT commands

Command	Description
D *addr*	Sets the first word address of the entire display to address *addr*
D *leftaddr rightaddr*	Sets first word addresses on the left and right side of the display to left address *leftaddr* and right address *rightaddr*, respectively
DL *addr*	Sets the first word address on the left side of the display to address *addr*
DL *upperaddr loweraddr*	Sets the first word addresses on the upper and lower left side of the display to upper address *upperaddr* and lower address *loweraddr*, respectively
DLU *addr*	Sets the first word address on the upper left side of the display to address *addr*
DLL *addr*	Sets the first word address on the lower left side of the display to *addr*
DR *addr*	Sets the first word address on the right side of the display to *addr*
DR *upperaddr loweraddr*	Sets the first word addresses on the upper and lower right side of the display to *upperaddr* and *loweraddr*, respectively
DRU *addr*	Sets the first word address on the upper right side of the display to *addr*
DRL *addr*	Sets the first word address on the lower right side of the display to *addr*

CMOSX TEXT commands

CMOSX TEXT commands set the text format and first word memory addresses for the given quadrants and use the basic display FORMAT commands (see table 3-2). The default addresses for the text displays are 3600 for the Cray display and 17000 for the IOS display. Table 3-3 lists the CMOSX TEXT commands and briefly describes the function of each.

Table 3-3. CMOSX TEXT commands

Command	Description
DT *addr*	Displays text at address *addr* on the entire display
DT *leftaddr rightaddr*	Displays text at left address *leftaddr* and right address *rightaddr* on the left and right sides of the display
DTL *addr*	Displays text at address *addr* on the left side of the display
DTL *upperaddr loweraddr*	Displays text at upper address *upperaddr* and lower address *loweraddr* on the upper and lower left side of the display
DTLU *addr*	Displays text at address *addr* on the upper left side of the display
DTLL *addr*	Displays text at address *addr* on the lower left side of the display
DTR *addr*	Displays text at address *addr* on the right side of the display
DTR *upperaddr loweraddr*	Displays text at upper address *upperaddr* and lower address *loweraddr* on the upper and lower right sides of the display
DTRU *addr*	Displays text at address *addr* on the upper right side of the display
DTRL *addr*	Displays text at address *addr* on the lower right side of the display

CMOSX ROLL commands

CMOSX ROLL commands scroll the displayed memory forward or backward. Each targeted quadrant is rolled 40_8 parcels (10_8 words). Pressing the LINEFEED key clears the roll command. Table 3-4 lists the CMOSX display ROLL commands and briefly describes the function of each.

Table 3-4. CMOSX display ROLL commands

Command	Description
DF	Rolls all quadrants on the display forward
DLF	Rolls the left side of the display forward
DRF	Rolls the right side of the display forward
DB	Rolls all quadrants on the display backward
DLB	Rolls the left side of the display backward
DRB	Rolls the right side of the display backward

CMOSX parcel/word MODE commands

CMOSX parcel/word MODE commands toggle the display format between parcel and word mode. If you are in parcel mode and enter the M command, the entire display is toggled to word mode. If you are in word mode and enter the ML command, the left side of the display is toggled to parcel mode. Addresses cannot be designated for mode commands. Table 3-5 lists CMOSX parcel/word MODE commands and briefly describes the function of each.

Table 3-5. CMOSX parcel/word MODE commands

Command	Description
M	Toggles the entire display
ML	Toggles the left side of the display
MLU	Toggles the upper left quadrant of the display
MLL	Toggles the lower left quadrant of the display
MR	Toggles the right side of the display
MRU	Toggles the upper right quadrant of the display
MRL	Toggles the lower right quadrant of the display

CMOSX REFRESH control commands

CMOSX REFRESH control commands affect the way CMOSX refreshes displays. Table 3-6 lists CMOSX REFRESH control commands and briefly describes the function of each.

Table 3-6. CMOSX REFRESH control commands

Command	Description
:RE	Disables refresh until a LINEFEED is entered
:MU	Toggles the Micro Mutt flag; slowing the refresh rate to 1200 baud.
CTRL-Z	Clears and repaints the display (Press the CTRL and letter Z keys simultaneously.)
NULL ON	Sets the Null flag on (clears the screen and displays nulls)
NULL OFF	Sets the Null flag off
NULL	Toggles the Null flag

CMOSX DATE and TIME commands

The CMOSX DATE and TIME commands (table 3-7) can be used to set the date and time information that appears in the upper right quadrant of the display.

Table 3-7. CMOSX DATE and TIME commands

Command	Description
DATE *mm/dd/yy*	Enters the date (month, day, and year) on the display
TIME *hh:mm:ss*	Enters the time (hours, minutes, and seconds) on the display
DATE *mm/dd/yy* *hh:mm:ss*	Enters the Date and time on one line

CMOSX BASE ADDRESS command

The BASE ADDRESS command (BA) biases all address references by a base address. Base address is normally 0. If a base address other than 0 is selected, it is displayed on the top line of the screen. The base address affects all display, memory store, and file load commands. Set the base address by entering the following command:

BA *addr*

CMOSX FILE manipulation commands

FILE manipulation commands are used for manipulating files in the disk or tape library (see Appendixes A and B for a list of the diagnostics). The CMOSX file manipulation commands manipulate the files in the FNT associated with the current CMOSX subsystem, except where noted. Table 3-8 lists the FILE manipulation commands and briefly describes the function of each.

Table 3-8. CMOSX FILE manipulation commands

Command	Description
:LOAD *file fwa lwa* or /*file*	Reads *file* from the deadstart buffer at the address range specified. The default first word address (*fwa*) is 0; the default last word address (*lwa*) is 20000 for the IOS and 6000 for the CRAY X-MP system; the maximum size limits.
:GO *file linenum*	Loads and executes the command buffer *file* starting at line number *linenum* (optional)
:FILES	Displays the files in the library. While running under Subsystem 0 (:IO), FNT 3 files are displayed. While running under Subsystem 1 (:CR), FNT 4 files are displayed. All files are displayed while running under CMOSX/T.
:SAVE *file fwa lwa*	Writes *file* to the library from the deadstart buffer range specified. The default first word address (*fwa*) is 0; the default last word address (*lwa*) is 20000 for the IOS and 6000 for the CRAY X-MP system; the maximum size limits.

Table 3-8. CMOSX FILE manipulation commands (continued)

Command	Description
:DELETE *file*	Deletes program *file* from the library
:SCAN *file*	Displays the text file *file* on the console
:HELP	Displays FNT 5 files (HELP† documentation files) on the console
:HELP *file* or :file	Displays the HELP† file from FNT 5 on the console

† HELP is available on the disk system only.

CMOSX miscellaneous commands

Table 3-9 lists CMOSX miscellaneous commands.

Table 3-9. Miscellaneous CMOSX commands

Command	Description
:EX *s* *command*	Executes *command* on subsystem *s* where *s* can be 0, 1, or 2. The :EX command allows you to execute a command on the Subsystem 1 console from the Subsystem 0 console and so on. *command* can be any command.
:SN *comments*	Copies the display to the printer. *comments* is an optional parameter; any comments you want to print can be entered when it is specified.

3.1.5 CMOSX INTERNAL CONFIGURATION TABLES

CMOSX has Internal Configuration Tables that provide it with information on the hardware configuration of the system on which it is running. The configuration information must be given to CMOSX at the time it is installed on the site and is entered through a command buffer that prompts the user for the needed information when it is invoked. The information is then saved in a disk file on File Name Table (FNT) 3 which is loaded each time CMOSX is deadstarted.

System configuration (CONF) is entered on Subsystem 0 with the following procedure:

1. Boot CMOSX. See subsection 3.1.1.

2. Invoke the configuration command buffer, /CONF that brings up Subsystem 1 and configures CPU tests at your option by entering:

   :GO /CONF

   Upon completion, /CONF, saves a configuration file called CONDAT on FNT 3. Each time CMOSX is initialized, it reads the data from the file CONDAT into the appropriate system tables.

3. Reload (reboot) CMOSX after configuration to make sure you rewrite your IOP0 deadstart CMOSX/T so the tape system has the correct configuration file.

4. Mount the blank tape on the I/O Subsystem Peripheral Expander drive and while running under DSS, write the tape bootstrap as the first file on the blank tape by entering:

   WRITE TBOOT @

5. Write all of the files on tape directory file A0DIR on the blank tape by entering:

   WRITEA A0DIR

3.2 SUBSYSTEM 0 - I/O SUBSYSTEM (IOS) MAINTENANCE SYSTEM

Subsystem 0 runs diagnostics on the IOS and its associated peripherals. CMOSX runs in IOP0 and, depending on the test mode, any diagnostic or combination of diagnostics can be run simultaneously on IOP1 through IOP3. The minimum system hardware needed for CMOSX Subsystem 0 to load, to deadstart, and to monitor an IOS diagnostic is:

- IOP0 with Buffer Memory
- Peripheral Expander chassis with 80 Mbyte disk or tape drive

When a diagnostic is deadstarted, the following sequence of activities is initiated:

1. The diagnostic is loaded into IOP0 Local Memory from disk or tape at load time.
2. The diagnostic is written to Buffer Memory at deadstart time.
3. IOP0 electrically pulls (sets) the deadstart line for IOP*x* with a logical 1.
4. A Local Memory read of Buffer Memory is forced by IOPx.
5. The deadstart line is cleared electrically with a logical 0. When the deadstart line is clear, execution begins in IOP*x*.
6. The diagnostic monitor periodically writes all of IOPx Local Memory to Buffer Memory.
7. CMOSX continually reads selected addresses from IOPx Buffer Memory image and displays them on the console. The Subsytem 0 display is illustrated in figure 3-2.

Figure 3-3 illustrates steps 1 through 7.

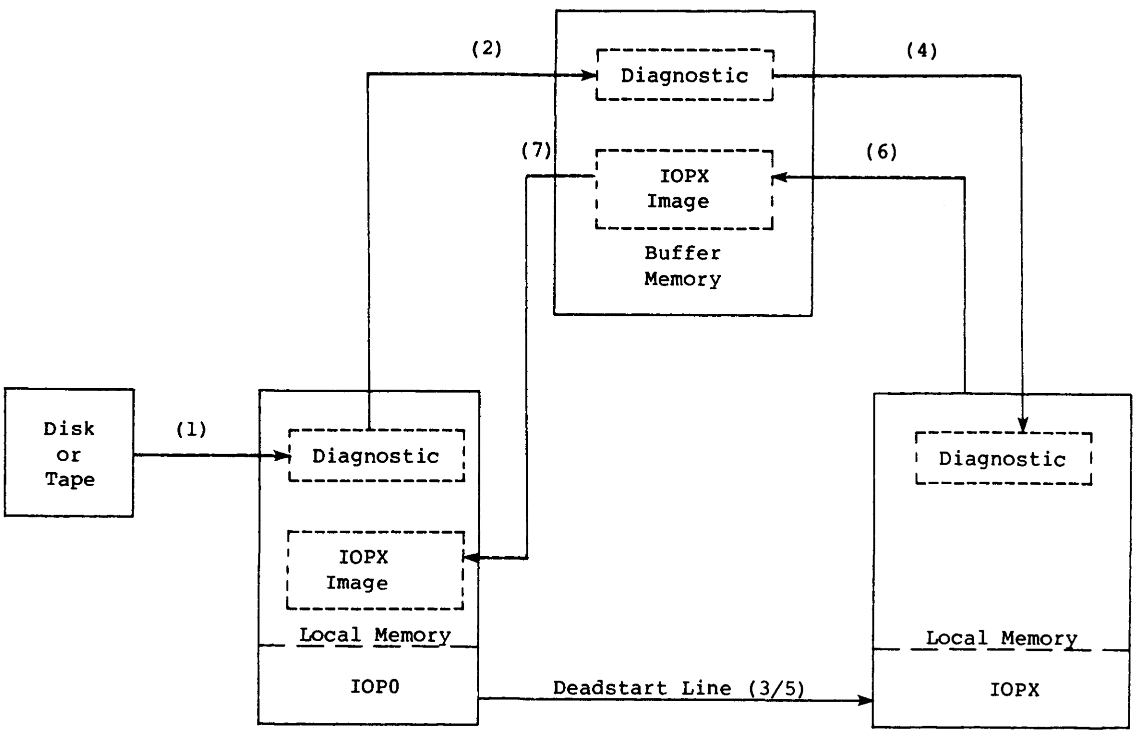


Figure 3-3. Loading and deadstarting a diagnostic in IOPx

Subsystem 0 commands are used to control displays, to execute diagnostics, and to print or store output. Subsystem 0 commands are only available when Subsystem 0 is running and include the following:

- Test MODE
- Current processor number
- DEADSTART/MASTER CLEAR
- Display FORMAT
  - Local Memory
  - Buffer Memory
  - MCU memory
- STORE memory
  - Local Memory
  - Buffer Memory
  - MCU memory
- DEBUGGING
- Miscellaneous

3.2.1 SUBSYSTEM 0 TEST MODE COMMANDS

Four test modes are available. The current mode is displayed on the top line of the screen. Use the appropriate mode to fit the situation. The default, Test Mode (:TM), can test all of the processors except IOP0. Table 3-10 lists the CMOSX Test MODE commands and briefly describes the function of each.

3.2.2 SUBSYSTEM 0 CURRENT PROCESSOR NUMBER COMMAND

The current processor number is indicated in the upper left corner of the display and determines the processor to be deadstarted or master cleared on a DS or MC function. In :TM mode, the current processor number also determines which processor's image area is to be displayed. Set the current processor number to *p* by entering the following command:

PN *p*

p can be 0 for IOP0, 1 for IOP1, 2 for IOP2, or 3 for IOP3.

Table 3-10. Subsystem 0 test MODE commands

Command	Description
:TM	Test Mode (:TM) multiprocessor. Tests any or all processors at once by allocating an image area in Buffer Memory for each IOP. :TM is the default test mode and requires either the simple (PTA) or advanced (PTI) monitor (Appendix B.2).
:TB	Test Basic (:TB). Tests one IOP at a time and requires either the PTA or PTI monitor (Appendix B.2).
:TD *delay wcount*	Test Dead (:TD). Repeatedly deadstarts *wcount* words into the selected processor at *delay* intervals. This mode is generally used when the monitor is not running or to scope an inoperative IOS. The default values are 10_8 for *delay* and 40_8 for *wcount*. The range of acceptable values is 1-7777 for *wcount* and 0000-17777 for *delay*.
:TS	Test mode self-test (:TS). Runs a diagnostic on IOP0 concurrently with CMOSX. The diagnostic executes out of the deadstart buffer and must have been assembled relocatable. The diagnostic can use only 20000 parcels of Local Memory (the size of the deadstart buffer) and only operand registers 600 and above.
:KI	Terminates the current mode (kills)

3.2.3 SUBSYSTEM 0 DEADSTART/MASTER CLEAR COMMANDS

DEADSTART/MASTER CLEAR commands, shown in table 3-11, are used to start or stop the running of a diagnostic in a particular processor. Different forms of each command can be used, depending on the test mode. When a diagnostic is loaded into the deadstart buffer, the diagnostic name appears in the upper right corner of the display.

When the system is deadstarted, the name appears just below. Two file names are displayed in :TM because different diagnostics can be running in each processor. The top file name is the diagnostic in the deadstart buffer. The bottom file name, signified by the greater than symbol (>), is the diagnostic running in the current processor.

Table 3-11. Subsystem 0 DEADSTART/MASTER CLEAR commands

Command	Description
DS	Deadstarts the current processor from the deadstart buffer
MC	Master clears the current processor
DS *p*†	Deadstarts processor number *p* from the deadstart buffer
DS A†	Deadstarts all processors from the deadstart buffer
MC *p*†	Master clears processor number *p*
MC A†	Master clears all processors

† Command supported by :TM mode only

3.2.4 SUBSYSTEM 0 DISPLAY FORMAT COMMANDS

Three types of memory can be displayed from Subsystem 0: Local Memory, Buffer Memory, and MCU memory. Both Local and Buffer Memory text-formatted displays are supported by CMOSX. Therefore, the LM and BM commands do not change the display format. The DT commands set (see text format) and the D commands clear text format. Conversely, the D and DT commands do not affect memory type.

Local Memory DISPLAY commands

Subsystem 0 Local Memory DISPLAY commands, shown in table 3-12, display the deadstart buffer from which diagnostics are deadstarted into other processors. Local Memory commands use the basic display command format.

Table 3-12. Subsystem 0 Local Memory DISPLAY commands

Command	Description
LM	Displays Local Memory on the entire display at the current address
LM *addr*	Displays Local Memory on the entire display at the given address
LM *leftaddr rightaddr*	Displays Local Memory on the left and right sides of the display at the given addresses
LML *addr*	Displays Local Memory on the left side of the display at the given address
LML *upperaddr loweraddr*	Displays Local Memory on the upper and lower left side at the given addresses
LMLU *addr*	Displays Local Memory on the upper left side of the display at the given address
LMLL *addr*	Displays Local Memory on the lower left side of the display at the given address
LMR *addr*	Displays Local Memory on the right side of the display at the given address
LMR *upperaddr loweraddr*	Displays Local Memory on the upper and lower right side at the given addresses
LMRU *addr*	Displays Local Memory on the upper right side of the display at the given address
LMRL *addr*	Displays Local Memory on the lower right side of the display at the given address

Buffer Memory DISPLAY commands

Subsystem 0 Buffer Memory DISPLAY commands display the image area in Buffer Memory that was written by the diagnostic's monitor (see table 3-13). Buffer Memory display commands use the CMOSX basic display command format.

Table 3-13. Subsystem 0 Buffer Memory DISPLAY commands

Command	Description
BML *addr*	Displays Buffer Memory on the left side of the display at the given address
BML *upperaddr loweraddr*	Displays Buffer Memory on the upper and lower left quadrants of the display at the given addresses
BMLU *addr*	Displays Buffer Memory on the upper left quadrant of the display at the given address
BMLL *addr*	Displays Buffer Memory on the lower left quadrant of the display at the given address
BMR *addr*	Displays Buffer Memory on the right side of the display at the given address
BMR *upperaddr loweraddr*	Displays Buffer Memory on the upper and lower right quadrants of the display at the given addresses
BMRU *addr*	Displays Buffer Memory on the upper right quadrant of the display at the given address
BMRL *addr*	Displays Buffer Memory on the lower right quadrant of the display at the given address

MCU memory DISPLAY commands

Subsystem 0 MCU memory DISPLAY commands display Local Memory for debugging CMOSX itself. Table 3-14 lists the Subsystem 0 MCU memory display commands and briefly describes the function of each.

Table 3-14. Subsystem 0 MCU memory DISPLAY commands

Command	Description
MM	Displays MCU memory on the entire display at the current addresses
MM *addr*	Displays MCU memory on the entire display at address *addr*
MM *leftaddr rightaddr*	Displays MCU memory on left and right sides of the display at left address *leftaddr* and right address *rightaddr*
MML *addr*	Displays MCU memory on the left side of the display at address *addr*
MML *upperaddr loweraddr*	Displays MCU memory on the upper and lower left quadrants of the display at upper address *upperaddr* and lower address *loweraddr*
MMLU *addr*	Displays MCU memory on the upper left quadrant of the display at address *addr*
MMLL *addr*	Displays MCU memory on the lower left quadrant of the display at address *addr*
MMR *addr*	Displays MCU memory on the right side of the display at address *addr*
MMR *upperaddr loweraddr*	Displays MCU memory on the upper and lower right quadrants of the display at upper address *upperaddr* and lower address *loweraddr*
MMRU *addr*	Displays MCU memory on the upper right quadrant of the display at address *addr*
MMRL *addr*	Displays MCU memory on the lower right quadrant of the display address *addr*

3.2.5 SUBSYSTEM 0 STORE MEMORY COMMANDS

Subsystem 0 STORE memory commands store data in Local Memory, Buffer Memory, and MCU memory.

Local Memory STORE commands

Table 3-15 lists the Subsystem 0 Local Memory STORE commands and briefly describes the function of each.

Table 3-15. Subsystem 0 Local Memory STORE commands

Command	Description
S *addr data*	Stores *data* in Local Memory (deadstart buffer) at address *addr*
S+*addr data*	Stores *data* in Local Memory (deadstart buffer) at address *addr* and increments the pointer to the next address
S *addr data*0 *data*1 *data*2 *data*3	Stores up to 4 octal parcels provided address *addr* is on a word boundary

Buffer Memory STORE commands

Table 3-16 lists the Subsystem 0 Buffer Memory STORE commands and briefly describes the function of each.

Table 3-16. Subsystem 0 Buffer Memory STORE commands

Command	Description
SB *addr data*	Stores *data* in Buffer Memory (image area) at address *addr*
SB+*addr data*	Stores *data* in Buffer Memory (image area) at address *addr* and increments the pointer to the next address
SB *addr data*0 *data*1 *data*2 *data*3	Stores up to 4 parcels provided address *addr* is on a word boundary

MCU memory STORE command

The MCU memory (SM) STORE command is used for patching and debugging the CMOSX system. It stores *data* in MCU Memory at *addr* and is entered in the following way:

SM *addr data*

3.2.6 SUBSYSTEM 0 DEBUGGING COMMAND

The Subsystem 0 DEBUGGING command sends an image of IOP0 operands A, B, and C, the operand registers, and exit stack to MCU memory (currently 106000) at address *addr* while code continues execution. The contents of the operating registers can then be examined using the MCU memory DISPLAY commands. See table 3-14. The debugging command is entered as follows:

:XD *addr*

3.2.7 SUBSYSTEM 0 MISCELLANEOUS COMMANDS

Table 3-17 lists the miscellaneous commands for Subsystem 0 and briefly describes the function of each.

Table 3-17. Subsystem 0 miscellaneous commands

Command	Description
:DP	Dumps the processor being tested to its Buffer Memory image area. Specify the desired test mode to return to after using this command.
:RB	Replaces the current deadstart buffer contents with the current image of the test from Buffer Memory
:HI	Starts a routine that enables error information to be sent from IOP0 to IOP1. :HI is used in only a few high-speed channel tests and puts Subsystem 1 in high-speed mode; stops Cray diagnostics.

3.3 SUBSYSTEM 1 - CRAY CPU MAINTENANCE SYSTEM

Subsystem 1 runs diagnostics on a Cray CPU. It uses the 6 Mbyte (low-speed) channel from IOP0 for all transfers to and from the CPU. The minimum hardware needed for CMOSX Subsystem 1 to load, to deadstart, and to monitor a CPU diagnostic is:

- IOP0
- Peripheral Expander chassis with 80 Mbyte disk or tape drive
- 6 Mbyte channel; both CPU and IOP0 sides.

Figure 3-4 is an example of the Subsystem 1 display.

```
MODE TB                          CMOSX/I-C  1.0       00/00/00        *EXJ
                                                      00:09:45

C X 0120                               C
C                                      C
   P  00001313-0  XA 0160  A0 00000000   20 000000 000000 000000 000000
   BI 00132540    VL  100  A1 00000000   21 000000 000000 000000 000000
   LI 00265340    VU    0  A2 00000000   22 000000 000000 000000 000000
   BD 00132540    PN    0  A3 00000100   23 000000 000000 000000 000001
   LD 00265340    PS    0  A4 00000000   24 000000 000000 000000 000000
   FL     0 041   CN    0  A5 00000000   25 000000 000000 000000 005454
   MO    00  00   EM    0  A6 00000000   26 000000 000000 000000 000001
                           A7 00000000   27 000000 000000 000000 000000

   S0 000000 000000 000000 000000        30 000000 000000 000000 000000
   S1 000000 000000 000000 000000        31 000000 000000 000000 000000
   S2 000000 000000 000000 000000        32 000000 000000 000000 000000
   S3 000000 000000 000000 000000        33 000000 000000 000000 000000
   S4 000000 000000 000000 000000        34 000000 000000 000000 000000
   S5 000000 000000 000000 000000        35 000000 000000 000000 000000
   S6 000000 000000 000000 000000        36 000000 000000 000000 000000
   S7 000000 000000 000000 000000        37 000000 000000 000000 000000

ENTER  ?
```

Figure 3-4. Subsystem 1 display

Subsystem 1 commands are only available when Subsystem 1 is running and include the following:

- Test MODE
- DEADSTART/MASTER CLEAR

- Exchange Package DISPLAY
- STORE memory
- Exchange Package STORE
- FORMAT CPU Error table
- RUN

3.3.1 SUBSYSTEM 1 TEST MODE COMMANDS

Table 3-18 lists the Subsystem 1 test MODE commands and gives a brief description of the function of each.

Table 3-18. Subsystem 1 test MODE commands

Command	Description
:TB	Test Basic (:TB). Assumes CPU 6 Mbyte, memory, exchange, and basic instruction issues are working. :TB is the default test mode.
:TD *delay wcount*	Test Dead (:TD). Repeatedly deadstarts *wcount* words starting at address 0 and inputs *wcount* words starting at address 2000 at *delay* intervals. The default values are 10_8 for *delay* and 40_8 for *wcount*. The range of acceptable values is 0001-7777 for *wcount* and 0000-17777 for *delay*.
:DD *delay wcount*	Test Dead Dump (:DD). The same as :TD mode with one exception. Instead of using the normal channel to input display data, it uses the CPU dead-dump feature.
:RUN *file*	Test Run (:RUN). Deadstarts *file* into the CPU as the RUN monitor, and selects test RUN. The default for *file* is the RUN monitor.

3.3.2 SUBSYSTEM 1 DEADSTART/MASTER CLEAR COMMANDS

Subsystem 1 DEADSTART/MASTER CLEAR commands (table 3-19) start or stop the running of a diagnostic.

Table 3-19. Subsystem 1 DEADSTART/MASTER CLEAR commands

Command	Description
DS	Deadstarts the diagnostic from the deadstart buffer/image area into the CPU
MC	Raises the CPU master clear line which stops all CPU activity

3.3.3 SUBSYSTEM 1 EXCHANGE PACKAGE DISPLAY COMMANDS

The only display commands unique to Subsystem 1 are the Exchange Package DISPLAY commands (table 3-20). These commands display the contents of the Exchange Package on the console.

Table 3-20. Subsystem 1 Exchange Package DISPLAY commands

Command	Description
DX *addr*	Displays the Exchange Package on the left side of the display at *addr*
DX *addr*0 *addr*1	Displays the Exchange Package on the left and and right sides of the display at left address *addr*0 and right address *addr*1, respectively
DXL *addr*	Displays the Exchange Package on the left side of the display at address *addr*
DXR *addr*	Displays the Exchange Package on the right side of the display at address *addr*
DE	Displays CPU memory error information in the Exchange Package

3.3.4 SUBSYSTEM 1 STORE MEMORY COMMANDS

Table 3-21 lists Subsystem 1 STORE memory commands and briefly describes the function of each.

Table 3-21. Subsystem 1 STORE memory commands

Command	Description
S *addr data*	Stores *data* in address *addr*
S *addr data0 data1 data2 data3*	Stores *data*0 through *data*3 in address *addr*
S+*addr data*	Stores *data* in address *addr* and increments the pointer to the next address
S *addr-parcel data*	Stores *data* in the parcel addressed by *addr-parcel*
S+*addr-parcel data*	Stores *data* in the parcel addressed by *addr-parcel* and increments the pointer to the next parcel

3.3.5 SUBSYSTEM 1 EXCHANGE PACKAGE STORE COMMANDS

Subsystem 1 Exchange Package STORE commands (table 3-22) allow you to store addresses for Exchange Package displays and to reset Exchange Package mode bits, flag bits, and vector length values.

Table 3-22. Subsystem 1 Exchange Package STORE commands

Command	Description
BI *addr*	Sets the CPU instruction base address in the current Exchange Package to address *addr*
LI *addr*	Sets the CPU instruction limit address in the current Exchange Package to address *addr*
BD *addr*	Sets the CPU data base address in the current Exchange Package to address *addr*
LD *addr*	Sets the CPU data limit address in the current Exchange Package to address *addr*
P *addr*	Sets the current Exchange Package P to address *addr*

Table 3-22. Subsystem 1 EXCHANGE PACKAGE commands (continued)

Command	Description
M *data*	Sets the current Exchange Package mode bits to *data*
MX *data*	Sets extended mode bits to *data*
F *data*	Sets the current Exchange Package flag bits to *data*
V *data*	Sets the current Exchange Package vector length to *data*
X *addr*	Sets the current Exchange Package exchange address to address *addr*
CN *data*	Sets the current Exchange Package cluster number to *data*
A*n* *data*	Enters A register *n* in the current Exchange Package with *data*
S*n* *data*	Enters S register *n* in the current Exchange Package with *data*
S*n* *data0* *data1* *data2* *data3*	Enters S register *n* in the current Exchange Package with data that is in parcel format
S*n-p* *data*	Enters S register *n* in the current Exchange Package with only one parcel entered in parcel *p*

3.3.6 FORMAT CPU ERROR TABLE COMMAND

The FORMAT CPU Memory Error Table (FET) program formats diagnostic monitor addresses (400-577) and outputs them to the printer. To run the program, enter the following command:

```
:FET
```

3.3.7 RUN MONITOR

The RUN monitor is an interactive CPU monitor that allows the user to load, to run, and to monitor up to 16 diagnostics simultaneously. The RUN monitor interprets keyboard commands and provides job control for the diagnostics as they compete on the system. The internal control for individual diagnostics is provided by the MTX† monitor, which must be set in interrupt driven mode.

Keyboard commands are sent to the CPU first. If the CPU does not recognize the command, it is flagged for execution by CMOSX. Table 3-23 lists RUN monitor commands and gives a brief description of the function of each.

Table 3-23. Subsystem 1 RUN commands

Command	Description
LOAD *file*1 *file*2 *file*3 *file* 4 ... *file*20	Loads up to 20_8 diagnostics to the CPU
CLN *xx n*	Sets cluster in control point *n* to *xx*
TL *n*	Loads the diagnostics in test list number *n* to the CPU; test lists 1 through 3 are currently supported.
TLA *file*	Loads all control points with file *file*
START	Starts the execution of all diagnostics that were previously loaded
START *n*	Starts the execution of the diagnostic at control point *n*
STOP	Stops the execution of all diagnostics
STOP *n*	Stops the execution of the diagnostic at control point *n*
DROP	Drops the diagnostic at all control points
DROP *n*	Drops the diagnostic at control point *n*

† The MZ monitor is used instead of the MTX monitor with CRAY-1 Computer Systems.

Table 3-23. Subsystem 1 RUN commands (continued)

Command	Description
RESTART *n*	Starts the execution of the diagnostic at control point *n* at address 1000
SCAN ON	Reads all of CPU memory. If any memory errors occur, they can be seen on the error log display.
SCAN OFF	Turns memory scanning off

3.4 SUBSYSTEM 2 - ERROR LOG SYSTEM

Subsystem 2 is only used on an I/O Subsystem with an error log channel. Error log channels exist only on IOSs through serial number (S/N) 20.

The only commands allowed under Subsystem 2 other than those listed in the subsection are: :BO, :BY, :CR, :IO, :MU, :RE, CTRL-Z, :EX, and :SN.

The error log monitor continuously logs errors from Buffer Memory, CPU memory, the 100 Mbyte channel, and Local Memory (all processors). Each of the preceding generates a display and, in addition, a summary display of all of the error buffers is provided (see figure 3-5).

```
  CMOS V 1 ERROR LOG

  ERROR LOG SUMMARY

I/O PROCESSOR LOCAL MEMORY ERRORS      0

BUFFER MEMORY ERRORS                   0

HIGH SPEED CHANNEL ERRORS              0


ENTER  ?
```

Figure 3-5. Subsystem 2 display

Subsystem 2 display commands are listed in table 3-24.

Table 3-24. Subsystem 2 DISPLAY commands

Command	Description
DB	Displays Buffer Memory errors
DC	Displays CPU memory errors
DH	Displays 100 Mbyte channel errors
DL	Displays Local Memory errors
DS	Displays a summary of all errors
CA	Clears all error buffers
CB	Clears the Buffer Memory error buffer
CC	Clears CPU memory error buffer
CH	Clears the 100 Mbyte channel error buffer
CL	Clears the Local Memory error buffer
F	Scrolls the display forward one page†
B	Scrolls the display back one page†

† F and B have no affect on displays of only one page.

CPU BOOTS 4

4.1 DESCRIPTION

A CPU boot is a stand-alone program that performs tests on the CPU. When the CPU boot command (CPXM or CPUM) is invoked from the IOP0 console, the current program in control is replaced in IOP0.

4.2 CPXM†

CPXM (CRAY X-MP Computer Systems) and CPUM (CRAY-1 Models A, B, C, M, and S Computer Systems) are memory diagnostics that are similar in function and must be booted into IOP0. CPU memory is tested through a series of reads and writes over the 6 Mbyte channel. The CPU, itself, does not execute any instructions.

To run CPXM, select a test pattern and enable that test by entering the CPXM EXECUTION control P command (see table 4-3). Other commands are available to alter test selections, output displays, and program execution. Running CPXM or CPUM requires you to select the following:

- Tests to be enabled (table 4-1)
- Commands that control test displays and execution (table 4-2)

4.2.1 TEST SELECTION

Any of the nine tests described in table 4-1 can be selected.

† This manual always refers to the Central Memory test as CPXM. All references to CPXM are applicable to CPUM. Any differences between CPXM and CPUM are specifically noted in the text.

Table 4-1. CPXM test selections

Test	Size (words)	Description
0	2000_8†	Enters zeros into CPU memory and checks the to verify that it matches the input
1	2000_8†	Enters ones into CPU memory and checks the output to verify that it matches the input
2	2000_8†	Enters a pattern of ones and zeros into CPU memory and checks the output to verify that it matches the input
3	2000_8†	Enters a pattern (12525) into CPU memory and checks the output to verify that it matches the input
4	2000_8†	Presets all words to zeros. A word of ones is entered, and the information is read out and checked. The word of ones is moved to the next address and sent through again. This process is repeated until the first 2000_8 addresses are checked.
5	2000_8†	Presets all words to ones. A word of zeros is entered and the information is read out and checked. The word of zeros is moved to the next address and sent through again. The process is repeated until the first 2000_8 addresses are checked.
6	40_8††	Initiates a data path test similar to test 4 except that an individual bit is set to 1 and moved through a field of zeros
7	40_8††	Initiates a data path test similar to test 5 except that an individual bit is set to 0 and moved through a field of ones
8	2000_8	Initiates a test pattern that is selected by the user into CPU memory and checks the read out. Stores the test pattern between locations 20000_8 and 22000_8 before running the test

† 1000_8 words for CRAY-1 S and M Series Computer Systems
†† 20_8 words for CRAY-1 S and M Series Computer Systems

4.2.2 COMMANDS

CPXM control commands can be divided into two types: display and execution.

CPXM DISPLAY control commands

The CPXM display can be divided into halves: top and bottom. The top half of the screen provides general test information and is not altered by keyboard commands. This part of the CPXM display is shown in figure 4-1 and includes the following information:

(1)	Test number:	Number of the test being executed
(2)	Test enable:	Enabled test (any and all tests can be enabled and are run in the sequence entered) is denoted by an asterisk immediately following the test number
(3)	Total passes:	Total number of passes completed for all enabled tests
(4)	Total errors:	Total number of errors
(5)	Bank errors:	Word errors that have accumulated for each bank

```
                                MEMORY TEST
         (1)
TEST      0                  (2)                  (3) TOTAL  PASSES  000000
  0       1     2    3    4    5    6*   7*   8*      (4)           ERRORS  000000
         (5)
BANK ERRORS
          X0      X1      X2      X3      X4      X5      X6      X7
    0X  000000  000000  000000  000000  000000  000000  000000  000000
    1X  000000  000000  000000  000000  000000  000000  000000  000000
M
          -0      -1      -2      -3      -4      -5      -6      -7
000000  000000  000000  000000  000000  000000  000000  000000  000000
000001  000000  000000  000000  000000  000000  000000  000000  000000
000002  000000  000000  000000  000000  000000  000000  000000  000000
000003  000000  000000  000000  000000  000000  000000  000000  000000
000004  000000  000000  000000  000000  000000  000000  000000  000000
000005  000000  000000  000000  000000  000000  000000  000000  000000
000006  000000  000000  000000  000000  000000  000000  000000  000000
000007  000000  000000  000000  000000  000000  000000  000000  000000
```

Figure 4-1. CPXM memory display

Test results can be displayed on the bottom half of the screen in a number of different ways. You can select either memory or bit error information to be displayed on the bottom half of the CPXM display. When a memory display is selected, the memory address is shown in the far left column. Columns -0 through -3 display the output buffer in parcel format, and columns -4 through -7 display the input buffer in parcel format. Ten addresses are displayed at a time. If a bit error display is selected, one of the following displays must be selected from table 4-2:

- Total bit errors
- Bit versus bank errors

The header indicates the type of display. If the bit versus bank error display is being shown, the bank number also appears. Table 4-2 lists CPXM DISPLAY control commands and briefly describes the function of each.

Table 4-2. CPXM DISPLAY control commands

Command	Description
B	Display bank. Displays the total number of bit errors encountered thus far.
B *nn*	Bit versus bank errors. Displays bit errors according to bank origin, where *nn* is an octal number from 0 to 37.
D *nnn*	Display address. Displays the output versus the input buffer for the test, where *nnnn* is an octal number from 0 to 1770.
F	Find error. Stops for an error stop condition with memory displayed. The buffer is searched for the error; the search starts at whatever address is presently being viewed. If an error is found, the address that is in error is the first address displayed. If no error is found, the entry line is not cleared and address 0 is displayed.
<	Roll down. Displays the next lower-numbered bank or group of addresses
>	Roll up. Displays the next higher-numbered bank or group of addresses

CPXM EXECUTION control commands

CPXM EXECUTION commands not only enable and disable tests, but alter output displays and change the conditions under which tests execute as they are executing. Any command entered at the keyboard immediately overrides any previous command at the completion of the current test. Table 4-3 lists CPXM EXECUTION control commands and briefly describes the function of each.

Table 4-3. CPXM EXECUTION control commands

Command	Description
C	Continue. Allows a test to continue running after being interrupted by scope, loop on error, or error stop.
E†	Error stop mode. Executes when an error is encountered on a check. The test stops and remains idle until it is changed.
L†	Loop on error mode. Executes if an error is found on a check. The test continues on its current pass and performs input, output, and error checks. Loop on error mode remains active until the mode is changed.
N	None. Clears the scope, loop on error, and error stop command modes.
P $n_1, n_2 ... n_8$	P followed by a space and one or more test numbers enables the test associated with the number. The following command enables tests 1, 2, and 3: P 123 Test numbers can be entered in any sequence (P 312). You must enter the number of each test the operator wishes enabled. To enable tests 0 through 7, enter the following: P 01234567

† A letter appears on display right after the number of the test that is enabled. Upon encountering an error, an asterisk appears after the letter.

Table 4-3. CPXM EXECUTION control commands (continued)

Command	Description
P (continued)	The P command can be entered at any time and the most recent P command takes precedence. If a P 4567 command has been entered, and while test 5 is executing a P 123 is entered, test 1 begins executing immediately after the completion of test 5. The P command followed by no test numbers terminates testing after the current test has been completed.
P A	Enables tests 0-8
R	Restart. Clears all error and pass counters and immediately begins executing the enabled test.
S *addr data*	Store data. Stores *data* at address *addr*
S+ *addr data*	Store data and advance. Stores *data* at address *addr* and advances to the next address
X	Scope mode. Shuts down display and error checking and does an I/O transfer of data in the output buffer. The keyboard is active but not echoed. Only significant characters such as L (loop on error) or E (error stop) affect the display. Nonsignificant characters do not affect the keyboard.

ONLINE CPU TESTS 5

An online test is a diagnostic that runs online under the control of the customer's operating system; normally, the Cray Operating System (COS). The test can be written in CAL, CFT, or both. The following online tests are described in this section:

- CPU tests
- Tape test
- Disk tests
- Control programs

To run any of the tests described in this section, keypunch, or compose under a text editor, a job consisting of COS control statements for each test to be run and submit the job through the front-end computer. For more information about COS control statements, see the CRAY-OS Version 1 Reference Manual, publication SR-0011.

NOTE

Before the first run of online tests, the test binaries must be installed on the customer's operating system, using the system installation procedures described in the diagnostic release letter.

5.1 CPU TESTS

The following CPU tests can be run online. The diagnostic tests run concurrently with normal processing at low priority. Online CPU diagnostics always produce dump listings after termination, caused by time limit expiration or abort on CPU failure.

Name	Description
AHT	AH indexing test
ARB	A register basic
ARM	Address register

Name	Description
BRB	B register basic
CMX	Random instruction and operand test
MIT	Moving inversions memory test
SFA	Simulate floating-point add
SFM	Simulate floating-point multiply
SFR	Simulate floating-point reciprocal
SIS	Scalar instruction simulator
SR2	1- and 2-parcel instruction conflicts
SR3	Random instruction register conflicts
SRA	Scalar register add
SRB	Scalar register basic
SRL	Scalar register logical
SRS	Scalar register shift
SVC	Scalar and vector compare
TRB	T register basic
VPOP	Vector population count
VRA	Vector register add
VRL	Vector register logical
VRN	Vector random
VRR	Vector random with random length and increments
VRS	Vector register shift

A sample job that runs simulate floating-point add (SFA) is shown in figure 5-1.

```
JOB,JN=SFA,T=10,P=2,M=50.
ACCOUNT,AC=acctno.†
ACCESS,DN=SFA,ID=DIAGSYS.
MODE,FI=DISABLE.††
SFA.
EXIT.
DUMPJOB.
DUMP,FW=0,LW=500.
/EOF
```

Figure 5-1. Online CPU diagnostic job

† If required, your local account statement card is placed here.
†† The mode card is needed for any diagnostic that runs with floating-point interrupts disabled in stand-alone mode.

NOTE

Error status locations appear at addresses 170 through 200 for CPU diagnostic tests.

5.2 ONLINE TAPE TEST

The only available online tape diagnostic is LADDER. This test runs under the control of the Cray Operating System (COS) and executes as a normal user job, requesting a tape drive from the operating system. To run LADDER, a job (figure 5-2) consisting of COS control statements must be keypunched, or composed under a text editor, and submitted through the customer's front-end computer system.

A parameter on the job control statement establishes the dataset name to be used by the program. Other parameters are read in by the program from $IN and must follow the end-of-file control statement.† Each parameter is on a separate record, starts in column 1, and is right-justified.

Parameter	Description
1	Starting tape record length (5 digits)
2	Ending tape record length (5 digits)
3	Record length increment (+ for increasing and - for decreasing)
4	Read/write switch: 0 Write/read 1 Read 2 Write
5	Pass count (Number of times the program loop is executed; the default is 1.)

Figure 5-2 is an example of a COS job that runs the LADDER online tape diagnostic test.

† The end of the control statement file is designated by an end-of-file record.

```
JOB,JN=LADDER,T=100,M,*6250=1.
ACCOUNT,AC=acctno.†
ACCESS, DN=LADDER,ID=DIAGSYS.
ACCESS,DN=TAPE,DT=*TAPE,VOL=LADSC1,NEW,DF=IC,MBS=32000.
LDR,DN=LADDER,CNS.
LADDER,DIAG.
EXIT.
DUMPJOB.
DUMP,DSP,FL=405000,LW=420000.
/EOF
1
3500
100
0
50
2
/EOF
```

Figure 5-2. LADDER online tape JCL

5.3 ONLINE DISK TESTS

The online disk tests currently available are DDTEST and CMST.

5.3.1 DDTEST

DDTEST is a diagnostic that tests a spare DD-19, DD-29, DD-39, or DD-49 disk drive, the Solid-state Storage Device (SSD), or the Block Multiplexer (BMR) online. To run the DDTEST, do the following:

1. Read the DDTEST test procedure.
2. Specify the test conditions on the DDTEST control statement.
3. Set the DDTEST sense-switches (table 5-1).
4. Run DDTEST (figure 5-3) using COS JCL.
5. Determine how the program terminates.
6. Display error information, if any, on the screen.

† Replace this statement with your local account statement. If your site does not use an account statement, delete the line.

Test procedure

When the parameter list has been entered (see format) and the sense-switches (table 5-1) have been set, follow the progress of the test using the procedure described below. The program reads the parameters passed to it and decides what to do. In general, it does the following:

1. Takes the last four numeric characters of the DV parameter and appends them to the string ZZZ to produce a dataset name.

2. Accesses a dataset with the dataset name from step 1 or with the name specified by the DN parameter.

3. If the file exists and the DELETE parameter is present, the program deletes and releases the dataset.

4. If the file does not exist or was deleted, it is assigned by the program to the specified device.

5. DDTEST goes into wait status, until sense-switch 2 is set, giving you the opportunity to switch on the disk being tested (normally at most sites it would be switched off). If the TRIAL parameter is specified, the program does not wait.

6. When you set sense-switch 2, the program starts immediately. The program writes/overwrites the ZZZDV file with the next test pattern, one track per record.

7. On the first iteration, the program saves the ZZZDV file, unless the NOSAVE parameter was specified. If the dataset is already permanent, the program calls adjust instead.

8. Depending upon which tests were required, the program reads the dataset sequentially, in oscillatory or random mode. Oscillatory mode reads the first track, the last track, the second track and so on, maximizing disk head movement.

9. Step 8 is repeated until the required number of iterations (six to eight) are exhausted, or until you terminate the program by setting sense-switch 1.

10. If any errors occur, they are reported in the logfile. Data check or block-number errors are recorded in the $OUT file.

    When an error is detected, the track on which the error occurred is flagged internally as bad and is retried. If at any time, on the retry or on a different test, that same track gives a second error, the track is internally flawed and is used again during a run of the DIDK test utility.

11. You have the option of running tests on up to eight disk drives. These tests are not performed in parallel. The iteration of tests is performed first on one drive, then on the next, and so on.

12. You have the option of using blocked or unblocked I/O. The default is unblocked, because it cuts down on memory (no I/O buffers are required).

    COS library routines are used to identify the track that is in error. The use of SETPOS† is asynchronous, and since no read ahead is performed, a bad status from the COS UNIT† I/O status routine indicates that the error is in the track being read.

    Since there is no equivalent of a block-number check with an unblocked dataset, the test is done internally by setting the last word of the track (word 22000B) to the value of the track number. This test is only run when the DDTEST data format parameter is specified as unblocked (DF=U). The last word is tested against the track number and any discrepancy causes a block-number error to be reported.

Control statement

The format for the DDTEST control statement is as follows:

DDTEST,DELETE,DELINT=*hh:mm:ss*,DELLEN=*hh:mm:ss*,DF=*format*,DN=*filename*,
DT=*devicetype*,DV=DD-*nn*-*nn*,LOOP=*count*,MSG=*msgtype*,TEST=*test*,NOSAVE,
NTKS=*nnnn*,NOACC,NOENG,NODELAY,PATTYPE=*type*,PERCENT=*nn*,RANSEED=*num*.

DDTEST parameters are in keyword format and are explained below.

† COS library routine. For more information about COS library routines, see the Library Reference Manual, CRI publication SR-0014.

DELETE — The program usually accesses the ZZZDV dataset before overwriting it. The DELETE parameter deletes and releases the dataset first; the default is no parameter.

DELINT — The amount of time that the program executes before going into DELAY state. Sense-switch 2 must be set for this parameter to take effect. The format of this parameter is variable: it can be *ss*, *mm*:*ss* or *hh*:*mm*:*ss*. For example:

DELINT=30 means 30 secs.
DELINT=1:30 means 1 min 30 secs.
DELINT=1:1:30 means 1 hr 1 min 30 secs.

The maximum value is equal to 5:00:00; the default is DELINT=1:0.

DELLEN — The amount of time that the program goes into DELAY state. Sense-switch 2 must be set before this parameter can be specified. The format is variable: it can be *ss*, *mm*:*ss*, or *hh*:*mm*:*ss*. For example:

DELLEN=30 means 30 secs.
DELLEN=1:30 means 1 min 30 secs.
DELLEN=1:1:30 means 1 hr 1 min 30 secs.

The maximum value is equal to 5:0:0; the default is DELLEN=1:0.

DF=*format*

The data format of the ZZZDV dataset. The data can be formatted in any of the following ways:

DF=U Default; the file is or will be unblocked.
DF=B The file is or will be blocked.

If an unblocked format is used, the last word of every track contains the track number to check for a block number error in a blocked dataset.

DN=*filename*

The file name to be used for testing the device. If this parameter is not specified (default), the name is generated internally according to the description given for the DV parameter. For example, the following command names three files:

DN=TESTDS1:TESTDS2:TDS3

DT=*dt* The device type of the device being tested. *dt* can be one of the following:

DD19	DD-19 disk drive
DD29	DD-29 disk drive
DD39	DD-39 disk drive
DD49	DD-49 disk drive
SSD8	8 Mword Solid-state Storage Device (SSD)
SSD	16 Mword SSD
SSD16	16 Mword SSD
SSD32	32 Mword SSD
BMR	1 Mword BMR
BMR1	1 Mword BMR

The default setting for the DT parameter is unspecified (UNDEF). If the DT parameter is UNDEF, the DV parameter is scrutinized, and if it starts DD-19-xx OR DD-29-xx, the DT parameter is assumed to be undefined as shown in the example below:

```
DT=UNDEF:UNDEF: ... :UNDEF
```

DV=DD-*nn*-*nn*

The logical device name (LDV) of the disk drives to be tested. If the device name (DN=*filename*) parameter is not present, the device type and unit is appended to the string ZZZ (ZZZ1930) to produce the name of the dataset that tests the disk (DV=DD-19-30). For example:

```
DV=DD-19-32:DD-29-62:DD-19-53
```

LOOP=*n* Number of iterations to perform; the default is 9999999. An iteration consists of writing a single pattern and performing the requested tests. Entering the following statement runs 11 data path patterns:

```
LOOP=11
```

MSG=*msgtype*

The type of messages, if any, that are sent to the user and system logs are specified by this parameter; the default is MSG=ALL.

NONE	Only error information messages
ALL	Progress messages and error information

NOSAVE The program usually saves the ZZZDV dataset after it has been written, unless it is already permanent. The NOSAVE parameter prevents this. The default is no parameter.

NTKS=*n* Number of tracks to be tested for each disk; the default is the complete disk. For example:

```
NTKS=:2000
```

If there are multiple DVs, each one can have a different NTKS value. If there is no specification for a certain DV, the value previously specified is used. The default is a complete disk. For example:

DD-19	4109 tracks are tested
DD-29	8219 tracks are tested
DD-39	4202 tracks are tested
DD-49	7086 tracks are tested
BMR	83 tracks are tested
SSD-8	908 tracks are tested
SSD16	1818 tracks are tested
SSD32	3636 tracks are tested

The number of tracks reserved for you is subtracted from the default value unless NOENG (see below) is specified. Once a value is specified, it is used for any succeeding devices for which a value is not specified.

NOACC The program is prevented from attempting to access the ZZZDV dataset. NOACC is useful for testing when several copies of DDTEST are being run on the same device. Instead of having to queue for access to the ZZZDV dataset, the tests run without it. The default is no parameter.

NOENG The disk drive under test does not have any tracks reserved for you. This rarely is the case, since seven cylinders are always reserved by COS for you, and COS itself requires modification to release engineering tracks. The NOENG parameter is available in case modifications are necessary. The default is no parameter.

NODELAY The program is prevented from going into DELAY mode. The program starts executing without waiting for you to switch on the disk drive to be tested. The job comes out of DELAY state immediately. NODELAY works as if you switched sense-switch 2 on and off. The default is no parameter.

PATTYPE=*type*
The number of the pattern to be used for testing the disk drive; the default is PATTYPE=ALL. Valid values for PATTYPE are:

ALL All the assembled and randomly generated patterns are used.

RANDOM

Randomly generated patterns are used for testing (1-9999). The pattern that is normally used on the current iteration when ALL is specified is used. In the following example, the 6th pattern is used:

mod[18,12])

PERCENT Percentage of the number of tracks (NTKS) to be tested; the default is PERCENT=100. The actual number of tracks to be tested is calculated using the following formula:

NTKS*PERCENT/100.

The formula allows for specifying the number as a percentage of a disk. The value given applies to any succeeding devices in which the percentage is not specified. The percentage to be tested is specified for two devices in the following example:

PERCENT=10:50

RANSEED=*n*

Seed that generates the sequence of random numbers used for the random pattern. RANSEED is specified as an integer in the range from 0 through 99999; the default is RANSEED=0.

TEST=*test*

The following tests can be performed:

SR Sequential read
OR Oscillatory read
RR Random read

The default is TEST=SR. The tests can be run one after another by listing them as shown in the example below:

TEST=SR:OR:RR

TRIAL The parameter that allows new program features to be tested. TRIAL causes the program to simulate both I/O and data validity errors. The default is no parameter.

Sense-switches

DDTEST sense-switches are software switches that contol the operation of the DDTEST program during execution. The default setting for all sense-switches is OFF. Table 5-1 lists DDTEST sense-switch settings and briefly describes the function of each during DDTEST execution.

Table 5-1. DDTEST sense-switches

Sense-switch	Significance
SW-1	Stops the program; if sense-switch 1 is on, the program stops at the end of the current test.
SW-2	Delays program execution so you have time to switch on the disk to be tested. Once the disk is switched on, sense-switch 2 should be switched on (it is normally configured off) to allow the program to continue. Sense-switch 2 also regulates the speed at which the program proceeds. If it is switched on, it causes the program to periodically go into WAIT TIMED-EVENT state preventing the program from taking over complete control of a disk channel. If it is not on, no delays occur.
SW-3	Restricts the tests to be performed to SR only. Sense-switch 3 enables one to change from OR and/or RR testing to sole SR testing. Without sense-switch 3 the program would have to be rerun with different parameters. Setting the switch off causes the program to revert to the tests specified by the TEST parameter.
SW-4	Restricts the data pattern used to RANDOM only. When sense-switch 4 is enabled, the test pattern changes from the current pattern to RANDOM. Without sense-switch 4, the program would have to be rerun with different parameters. Setting the switch off causes the program to revert to the patterns defined by the PATTYPE parameter. The default has none of the sense-switches set.

Program termination

If less than 100 error tracks are reported, DDTEST terminates at test completion. However, when 100 error tracks have been reported, the maximum limit is reached and the program aborts. Tracks that were reported to be in error are retried. If sense-switch 2 is set, the program goes into wait state. The length of the delay is determined by the DELINT and DELLEN DDTEST parameters .

Error information

To monitor control of the program, a common block pattern, which can be displayed using the COS DEBUG (see the CRAY-OS Version 1 Reference Manual, publication SR-0011) command, has been set up. To display this block, enter the following commands:

```
DIS  A  200  W  J  jobsequencenumber
DIS  B  220  W  J  jobsequencenumber
AB.
```

The DDTEST error summary is only displayed when the program identifies bad disk tracks that were not recognized by the hardware. The error summary displays the number of the track in error, the number of errors, and the type of error.

Two types of disk errors are recognized by DDTEST: picks (1 for a 0) and drops (0 for a 1). If inconsistent or intermittent errors are reported by DDTEST, check for bad data. Figure 5-3 is an example of the job control language (JCL) that runs DDTEST and displays the DDTEST error summary following a job abort.

```
JOB,JN=ABC,M=60,T,US=XYZ.
ACCOUNT,AC=xxxxxxx,US=Uyyyy,UPW=Uyyyy.†
RELEASE,DN=$IN.
DISPOSE,DN=$OUT,DC=SC.††
ASSIGN,D4=$OUT,BS=1,DC=PRT.††
ACCESS,DN=DDTEST,ID=DIAGSYS,OWN=Uzzzz.††
DDTEST,NOACC,NOSAVE,DELETE,NODELAY,NTKS=1760,DV=DD-A2-33,
       DT=DD49,LOOP=100,TEST=SR:OR:RR.
EXIT.
DUMPJOB.
ACCESS,DN=$DEBUG,PDN=DDTESTDEBUG.
DUMP,JTA,CENTER,FW=0,LW,DSP,V.
DEBUG,BLOCKS,TRACE.
*.
*.    DDTEST FAILED.....CALL AN ENGINEER.
*.
```

Figure 5-3. DDTEST JCL example

An example of the error summary generated by DDTEST is shown in figure 5-4.

† If required, your local account statement card is placed here.
†† These statements keep the number of required buffers at a minimum.

```
   ERROR SUMMARY

DV=DD-A1-32,  ERROR NO=1,  TIME=10:59:35,  TYPE=D,  RETRY COUNT=0,  TRACK NO=53

        TEST PATTERN= 0405004477331652072117B
        READ PATTERN= 0405004477331052072117B
        DIFFERENCES   0.........1.........2.........3.........4.........5.........6... ↑---BIT
        +PICK / - DROP                                       --

DV=DD-A1-32,  ERROR NO=2,  TIME=10:59:38,  TYPE=D,  RETRY COUNT=0,  TRACK NO=1696

        TEST PATTERN= 0405004477331652072117B
        READ PATTERN= 0405004477331052072117B
        DIFFERENCES   0.........1.........2.........3.........4.........5.........6... ↑---BIT
        +PICK / - DROP                                       --

   DD-A1-32 - ERROR IN TRACK NO    53
   DD-A1-32 - ERROR IN TRACK NO  1696

- - - - - - - - - - - - - - - - - - - - -
- - DD-A1-32  NO OF ERRORS FOUND    2 - -
- -           NO OF TRACKS FLAWED=  0 - -
- - - - - - - - - - - - - - - - - - - - -

   END OF SUMMARY  -  JOB ABORTED.
```

Figure 5-4. DDTEST error summary

Figure 5-4. DDTEST error summary

5.3.2 CMST

CMST is an online disk test that runs under the Cray Operating System (COS). The diagnostic executes as a normal user job and requests its disk space from COS. A job consisting of COS control statements must be keypunched or composed under a text editor and submitted through the customer's front-end computer system.

Format:

CMST,DV=*logunitnum*,T=*section*,P=*pattern*,S=*buffersize*

CMST parameters are in keyword format and are listed below:

DV=*logunitnum*
Logical unit number of the COS device being tested

P=*pattern*
Test pattern; *pattern* can be one of the following:

0 All zeros
1 All ones
2 Checkerboard
3 Word index and block index
4 Complement word index and block index

S=*buffersize*
Size of disk buffer in 512-word blocks

T=*section*
Test section; *section* can be one of the following:

0 Run all sections
1 Write sequential
2 Read sequential
3 Random read
4 Random write
5 Aggressor file

Figure 5-5 is an example of a COS job that runs the CMST test.

```
JOB,JN=CMST,T=100.
ACCOUNT,AC=acctno.†
ACCESS(DN=CMST,ID=DIAGSYS).
LDR,DN=CMST,CNS.
CMST,DV=A1-19-31,T=0,P=0,S=1.
/EOF
```

Figure 5-5. Online disk test (CMST) job

† If required, your local account statement card is placed here.

5.4 CONTROL PROGRAMS

The Diagnostic Systems Department (DSD) currently supports the following online control programs:

- Diagnostic sequencer (DSEQ)
- MENU utility

5.4.1 DIAGNOSTIC SEQUENCER (DSEQ)

The diagnostic sequencer (DSEQ) allows a list of online diagnostics to be run in a prescribed order. Each diagnostic has an associated pass count that can be modified. The sequencer operates in the following manner:

1. When control passes from the sequencer to a diagnostic, the diagnostic exchanges to the operating system to pick up the number of passes to be run.
2. When the specified pass count has been reached, the diagnostic exchanges to the operating system again.
3. DSEQ regains control from the completed diagnostic and does one of the following:
   a. Passes control to the next diagnostic on DSEQ's list
   b. Goes into a delay. When the delay has expired, DSEQ starts at the beginning of the list again.

DSEQ allows you to change the pass count and delay for individual diagnostics. For example, to modify the pass count from 100 to 200 passes, change DSEQ's job control language (JCL) from:

```
EXECUTE,VRN,PASS=100.
```

to the following:

```
EXECUTE,VRN,PASS=200.
```

Delay is a subroutine that is called by the DSEQ's JCL. To modify the delay period, it is necessary to edit the subroutine that determines the delay period.

DSEQ can be initiated using the MENU utility (figure 5-7) or submitted like any normal batch job (the sequencer is made up of JCL statements and procedures) to the Cray Operating System (COS).

5.4.2 MENU UTILITY

Online diagnostics can be selected using the interactive MENU utility. To run diagnostics using this utility from the I/O Subsystem (IOS), do the following:

1. Initialize an interactive console as described in the I/O Subsystem (IOS) Operator's Guide, CRI publication SG-0051.

2. Enter the following COS control statements at the interactive console:

   ```
   /LOGON
   ACCOUNT,AC=accntno.†
   ACCESS,DN=MENU,ID=DIAGSYS.
   MENU.
   ```

3. The following messages are displayed at the interactive console:

   ```
   CRAY ONLINE DIAGNOSTICS
   ENTER ACCOUNT STATEMENT
   ```

4. Respond to the console messages by entering the account statement that is required at your site.

5. When the account statement has been accepted, the following message is displayed at the interactive console:

   ```
   ENTER PERMANENT DATASET ID
   ```

6. Respond to the request by entering the following:

   ```
   DIAGSYS
   ```

7. MENU's first menu is displayed. See figure 5-6.

8. Select an option from the first menu by entering the number associated with the option. Options 1 through 8 cause various groups of online diagnostics to be submitted. Option 0 terminates the menu program.

9. Select option 9 from the first menu to display the MENU's second menu. See figure 5-7.

---

† If required, your local account statement card is placed here.

10. The second menu allows you to select the online sequencer (DSEQ) and the individual diagnostics listed on the screen. To enter any test, enter the number of the test, and press the RETURN key. CPU diagnostics submitted from the MENU utiltiy are terminated when a specified time limit is exceeded. Passing tests discard their output files.

11. To terminate the second menu and return to the first menu, enter 0.

12. To terminate the online MENU utility, enter 0 whenever the first menu is displayed. The following message is displayed on the console when the MENU utility has been terminated:

```
NOW RETURNING TO INTERACTIVE CONCENTRATOR
```

```
  AVAILABLE DIAGNOSTIC GROUPS
==============================

  1) ADDRESS REGISTERS
        - AHT, ARB -
  2) B AND T REGISTERS
        - BRB, TRB -
  3) SCALAR REGISTERS & FUNCTIONAL UNITS
        - SIS, SR3, SRA, SRB, SRL, SRS, SVC, CMX -
  4) VECTOR REGISTERS & FUNCTIONAL UNITS
        - VPOP, VRA, VRL, VRN, VRR, VRS, CMX, VCH -
  5) FLOATING POINT FUNCTIONAL UNITS
        - SFR, SFM, CMX, SFA -
  6) MEMORY TESTS
        - MIT -
  7) CPU CONFIDENCE TESTS
        -  SR3, CMX, VRN, VRR -
  8) ALL CPU DIAGNOSTICS
  9) SELECT INDIVIDUAL TESTS
        - INCLUDING CMST, DDTEST, LADDER, AND SEQUENCER -

  ENTER SELECTION NUMBER, OR 0 TO EXIT
```

Figure 5-6. MENU utility display (first menu)

```
                 AVAILABLE DIAGNOSTICS
                ========================

  1) AHT               10) SR3               19) VRA
  2) ARB               11) SRA               20) VRL
  3) BRB               12) SRB               21) VRN
  4) CMX               13) SRL               22) VRR
  5) MIT               14) SRS               23) VRS
  6) SFA               15) SVC               24) CMST
  7) SFR               16) TRB               25) DDTEST
  8) SFR               17) VCH               26) LADDER
  9) SIS               18) VPO               27) SEQUENCER

  ENTER SELECTION NUMBER, OR 0 TO RETURN TO MAIN MENU
```

Figure 5-7. MENU utility display (second menu)

I/O SUBSYSTEM BOOTS 6

I/O Subsystem (IOS) boots are stand-alone tests that are booted into execution in IOP0 from DSS0 by using the BOOT command. (See part 1, section 3 for details on the BOOT command.) These tests include the following:

- Buffer Memory test (BMT)
- DUMP
- I/O Processor Memory test (IOPM/IOPMA)
- Inter-processor channel boot (IPC)
- Multiprocessor Buffer Memory boot (MBUF)
- Basic checkout display (BCD) †
- Basic processor instruction test (BPX) †
- Disk system flaw test (DKX) †
- Save a zero test (SAZ) †

6.1 BUFFER MEMORY TEST (BMT)

Buffer Memory test (BMT) tests Buffer Memory by performing a series of reads and writes using a specified test pattern. The actual results are then compared against the expected results (test pattern) to determine if the test has been successful. BMT parameters identify local site hardware, enable specific tests, and control execution.

Table 6-1 lists BMT parameters and briefly describes the function of each. To run BMT, set up the memory description parameter, bank select mask, and chip select mask. DKX sets up the rest of the parameters. The test has section 14 selected with processor 1 disabled and processors 2 and 3 enabled.

† This IOS boot is booted from DSS0 and exists on a separate deadstart tape.

To set any parameter, enter the bit pattern of the desired instruction in the corresponding address. For example, to enable the program to run on an 8-bank machine, enter *xxxxx0* at address 4001.

Figure 6-1 is a an example of the BMT display. Display commands can be entered as single characters (function keys or keyboard characters) or as addresses that store multiple characters. See figure 6-1.

```
KEYBOARD COMMANDS     BMT BUFFER MEMORY TEST    BK 0X 01234567
       SECTION 9   BLOCK WRITE                      1X
SINGLE CHARACTER ENTRY                          CHIP 0123  KE

RUBOUT        BACKSPACE AND ERASE
LINEFEED      DELETE KEYBOARD LINE
RETURN        PROCESS KEYBOARD LINE
ESCAPE        STOP DIAGNOSTIC
SPACE         START DIAGNOSTIC
O             DISPLAY COMMON MEMORY
L             DISPLAY LOCAL MEMORY
R             DISPLAY REGISTERS
N             BLANK DISPLAY
P             PARAMETER DISPLAY

MULTIPLE CHARACTER ENTRY

U ADDR          SET UPPER DISPLAY TO ADDR
L ADDR          SET LOWER DISPLAY TO ADDR
S ADDR DATA     STORE DATA AT ADDRESS
S+ADDR DATA     STORE SEQUENTIAL
```

Figure 6-1. BMT display

Table 6-1. BMT parameters

Address	Default	Description
4001	0	Memory description parameter *xxxxx0* 8 banks *xxxxx1* 16 banks *xxxx0x* 16K chips *xxxx1x* 64K chips *xxx0xx* 128K memory *xxx1xx* 256K memory *xxx2xx* 512K memory *xxx4xx* 1024K memory

Table 6-1. BMT parameters (continued)

Address	Default	Description
4001		*xx* 1 *xxx* 2048K memory *xx* 2 *xxx* 4096K memory *xx* 4 *xxx* 8192K memory 0 *xxxxx* Section 0 1 *xxxxx* Section 1
4002	0	*xxxxx* 0 16K chip (2MA) *xxxxx* 1 64K chip (2MK)
4003	77577	*xxxxx* 1 section 0 Zeros *xxxxx* 2 section 1 Ones *xxxxx* 4 section 2 Sliding 1 *xxxx* 1 *x* section 3 Sliding 0 *xxxx* 2 *x* section 4 CS/BK ones *xxxx* 4 *x* section 5 CS/BK zeros *xxx* 1 *xx* section 6 Bank address *xxx* 2 *xx* section 7 Chip address *xxx* 4 *xx* section 8 Random data *xx* 1 *xxx* section 9 Block write *xx* 2 *xxx* section 10 Block read *xx* 4 *xxx* section 11 LM address *x* 1 *xxxx* section 12 SECDED *x* 2 *xxxx* section 13 Random R/W *x* 4 *xxxx* section 14 Random read 1 *xxxxx* section 15 Error channel
4004	15	*xxxxx* 1 Stop on error *xxxxx* 2 Stop at end of test *xxxxx* 4 Stop on SECDED error *xxxx* 1 *x* Stop on data error
4005	0	max address 1777 512K 3777 1024K 17777 4096K 37777 8192K
4006	0	*xxxx* 10 8 banks *xxxx* 20 16 banks
4007	177777	Bank select mask 000001 Bank 0 000002 Bank 1 000377 Bank 0, 1, 2, ... 7 177400 Bank 10, 11, ... 17

Table 6-1. BMT parameters (continued)

Address	Default	Description
4010	17	Chip select mask *xxxxx*1 Chip 0,1 *xxxxx*2 Chip 0,1,2 *xxxxx*3 Chip 0,1,3
4012	5	Buffer Memory data channel
4013	16	Buffer Memory error channel
4014	1	Quick look flag
4015	1000	Word count limit (sections 9,10)
4016	0	Refresh wait count
4017	177770	Loop count limit (section 13)
4020	6	Section 14 processor selects 1 A1 2 A2 4 A3
4021	0	Background Memory limit maximum address 1777 512K 3777 1024K 17777 4096K
4022	0	Northstar on system 1 Yes 0 No

6.2 <u>DUMP</u>

The DUMP boot allows the contents of Local Memory, Buffer Memory (MOS), or CPU (Central) memory to be listed on the printer. To produce a dump, answer Y (yes) or N (no) to several questions and specify a beginning and ending address for the dump. Figure 6-2 is an example of the DUMP display. To terminate a dump in progress, enter the following command:

/LF

When a dump is made of Buffer Memory or CPU memory, a listing is generated. To determine the format of the listing, enter either WORD or PARCEL next to the following screen prompt:

WORD MODE ?

You are also requested to enter a BASE ADDRESS when CPU Memory is dumped. When the system loads the DUMP program, it uses this address as the starting address of the program.

If IOP0 is to be dumped, it must have been saved previously by deadstarting the SAZ tape (subsection 6.9).

```
MOS?N
IOP-0?N
IOP-1?N
IOP-2?N
IOP-3?Y
LOCAL MEMORY:  0  10000

-LOCAL MEMORY
-OPERAND REGISTERS
-INTERNAL REGISTERS

MORE?
```

Figure 6-2. I/O Processor DUMP program

6.3 I/O PROCESSOR MEMORY TEST (IOPM/IOPMA)

IOPM tests memory in an I/O Processor. It tests the Local Memory of an elected processor by doing a deadstart/dead dump through Buffer Memory. IOPM is booted into execution in IOP0. One of the other processors can be selected to run the test, and any of nine test sections can be selected.

To run IOPM/IOPMA, select a test pattern and enable that pattern by entering the P command. Other commands are available to alter test selections, control output displays, and control program execution. Running IOPM/IOPMA requires you to select the following:

- Tests to be enabled (table 6-1)
- Commands that control test displays and execution (table 6-3)

6.3.1 TEST SELECTIONS

Any of the nine tests described in table 6-2 can be selected.

Table 6-2. IOPM/IOPMA test selections

Test	Size (parcels)	Description
0	4000_8	Enters zeros into the processor that is being tested to verify that it matches the input
1	4000_8	Enters ones into the processor that is being tested to verify that it matches the input
2	4000_8	Enters the 52525 pattern into the processor that is being tested and checks the output to verify that it matches the input
3	4000_8	Enters the 125252 pattern into the processor that is being tested and checks the output to verify that it matches the input
4	4000_8	Presests all parcels to zeros. One parcel of ones is entered and the information is read out and checked. The parcel of ones is moved to the next address and sent through again. This process continues until the first 4000_8 addresses are checked.
5	4000_8	Presets all parcels to ones. One parcel of zeros is entered and the information is read out and checked. The parcel of zeros is moved to the next address and sent through again. This process continues until the first 4000_8 addresses are checked.
6	100_8	Initiates a data path test similar to test 4 except that an individual bit is set to 1 and moved through a field of zeros.
7	100_8	Initiates a data path test similar to test 5 except a single bit is set to 0 and moved through a field of ones.
8	4000_8	Enters a user-selected test pattern into Buffer Memory and checks the read out. Stores the pattern using the S or S+ commands.

6.3.2 COMMANDS

IOPM/IOPMA control commands can be divided into two basic types: display and execution.

IOPM/IOPMA DISPLAY control commands

Figure 6-3 illustrates the IOPM display. The following information is included in the display:

(1)	Test number:	The number of the test being executed
(2)	Test enable:	An enabled test (any and all tests can be enabled and are run in the sequence entered) is denoted by an asterisk that immediately follows the test number
(3)	Total passes:	The total number of passes completed for all enabled tests.
(4)	Total errors:	The total number of errors accumulated for the test
(5)	Bank errors:	The word errors accumulated for each bank

```
                              MEMORY TEST
 (1)                                                             (3)
TEST  0         PROCESSOR  0                          TOTAL  PASSES   000000
  0   1   2   3   4   5   6*  7*  8* (2)                     ERRORS   000000
     (5)                                                       (4)
BANK ERRORS
         X0       X1       X2       X3       X4       X5       X6       X7
  0X  000000   000000   000000   000000   000000   000000   000000   000000
  1X  000000   000000   000000   000000   000000   000000   000000   000000

M
            -0       -1       -2       -3       -0       -1       -2       -3
000000   000000   000000   000000   000000   000000   000000   000000   000000
000001   000000   000000   000000   000000   000000   000000   000000   000000
000002   000000   000000   000000   000000   000000   000000   000000   000000
000003   000000   000000   000000   000000   000000   000000   000000   000000
000004   000000   000000   000000   000000   000000   000000   000000   000000
000005   000000   000000   000000   000000   000000   000000   000000   000000
000006   000000   000000   000000   000000   000000   000000   000000   000000
000007   000000   000000   000000   000000   000000   000000   000000   000000

ENTER ?
```

Figure 6-3. IOPM memory display

Test results can be displayed on the bottom half of the screen in a number of different ways. You can select either memory or bit error information to be displayed. When a memory display is selected, the memory address is shown in the far left column. Columns 2 through 5 (marked -0 through -3) display the output buffer in parcel format, and columns 6 through 9 (marked -4 through -7) display the input buffer in parcel format. If a bit error display is selected, one of the following displays must be selected from table 6-3:

- Total bit errors
- Bit versus bank errors

The header indicates the type of display. If the bit versus bank error display is being shown, the bank number also appears.

Table 6-3 lists IOPM/IOPMA DISPLAY control commands and briefly describes the function of each.

Table 6-3. IOPM/IOPMA DISPLAY control commands

Command	Description
B	Display bank displays the total bit errors encountered thus far.
B *n*	Bit versus bank errors displays errors according to bank origin, where *nn* is an octal number from 0 to 17
C	Continue allows a test to continue running after being interrupted by scope, loop on error, or error stop.
D *n*	Display address displays the input buffer for the test, where *nnn* is an octal number from 0 to 1770.
F	Find error stops for an error stop condition with memory displayed. The buffer is searched for the error. The search starts at whatever address is presently viewed. If an error is found, the address that is in error is the first address displayed. If no error is found, the entry line is not cleared and address 0 is displayed.
>	Roll down displays the next lower-numbered bank or group of addresses.
<	Roll up displays the next higher-numbered bank or group of addresses.

Table 6-3. IOPM/IOPMA DISPLAY control commands (continued)

Command	Description
:BOOT	Boots in another program
R	Restart clears all error and pass counters and immediately begins executing the enabled test.
:REFRESH	Disables the screen refresh until a LINEFEED is entered

IOPM/IOPMA EXECUTION control commands

IOPM/IOPMA EXECUTION control commands not only enable and disable tests, but alter output displays and determine the way tests execute before and even during execution. Table 6-4 lists and briefly describes the commands that control the execution of tests that run under IOPM/IOPMA.

Table 6-4. IOPM/IOPMA EXECUTION control commands

Command	Description
E †	Error stop mode is executed when an error is encountered on a check. The test stops and remains idle until it is changed by the operator.
L †	Loop on error mode is executed if an error is found on a check. The test continues on its current pass and performs input, output, and error checks. Loop on error mode remains active until it is changed by the operator.
N	The scope, loop on error, or error stop command is cleared by the NONE command.
P $n_1, n_2, \ldots n_8$	P followed by a space and one or more test numbers enables the test associated with the numbers. The following command enables tests 1, 2, and 3: P 123 command

† This letter appears on the display right after the number of the test that is enabled. Upon encountering an error, an asterisk appears after the letter.

Table 6-4. IOPM/IOPMA EXECUTION control commands (continued)

Command	Description
P (continued)	Test numbers can be entered in any sequence (P 312). The number of each test you want enabled must be entered. To enable tests 0 through 7, enter the following: P 01234567 The P command can be entered at any time and the most recent P command takes precedence. If a P 4567 has been entered, and while test 5 is executing P 123 is entered, test 1 begins immediately upon completion of test 5. When P is entered without test numbers, testing is terminated when the current test is completed.
P A	Selects patterns 0 through 8
PN *n*	Selects processor *n* for the test
X	Scope mode shuts down display and error checking and does an I/O transfer of data in the output buffer. The keyboard is active but not echoed. Only significant characters such as L (loop on error) or E (error stop) affect the display. Nonsignificant characters do not affect the keyboard.

6.4 INTERPROCESSOR CHANNEL BOOT (IPC)

The interprocessor channel boot (IPC) tests the interprocessor accumulator channels and checks data transfers that use IOD and IOB instructions, interrupts, deadstarts and dead dumps. The program assumes that Buffer Memory is running simple processor execution tests and working normally. IPC accepts execution parameters and returns error information at the completion of the test run.

6.4.1 PARAMETERS

IPC parameters can be used to select the test processor and set the program stop condition. Table 6-5 lists IPC parameters and their settings.

Table 6-5. IPC parameters

Address	Description
7	Test processor. Selection is made by setting bits in the following way: *xxxxx*1 Processor 0 *xxxxx*2 Processor 1 *xxxxx*4 Processor 2 *xxxx*1*x* Processor 3
10	Stop condition. Selection is made by setting bits in the following way: *xxxxx*1 Stop on error *xxxxx*2 Stop at end of test

6.4.2 ERROR INFORMATION

Errors are displayed in a table format that lists processor, channel, failing instruction, expected data, actual data, and the specific error condition (see figure 6-4). Errors generally occur in pairs. If the input processor finds a data error, it halts and reports it immediately. A halt causes the other processor in the pair to get an OUTPUT TIMEOUT error.

```
                                                  IPC INTER-PROCESSOR
                                                  CHANNEL TEST

                                                  PASS 000001
                                                  PROCESSOR PAIR

 PROC    CHAN    INSTR    EXPECT    ACTUAL    ERROR CONDITION
 ----    ----    -----    ------    ------    ---------------
```

Figure 6-4. IPC error display

6.5 MULTIPROCESSOR BUFFER MEMORY BOOT (MBUF)

Multiprocessor Buffer Memory (MBUF) test† is a system exerciser diagnostic that tests all of Buffer Memory from each processor individually and performs random writes and reads from all processors concurrently. MBUF performs an initial quick look with Buffer Memory dedicated, in turn, to each processor selected for the test. Essentially, a check is made of all addresses with reads and writes of zeros and ones.

Single word reads are interspersed throughout the test to compare against the random operation. Random block lengths up to 2000 words are used and testing continues indefinitely.

All interprocessor communications are handled over accumulator channels. A user-selected test pattern is stored by using the S or S+ command. Table 6-6 lists the MBUF test pattern commands and briefly describes the function of each.

Table 6-6. MBUF test pattern commands

Command	Description
S *addr data*	Stores *data* at *addr* in the write buffer. Data can be up to four parcels separated with spaces.
S+ *addr data*	Stores *data* at *addr* and advances entry to next address

During execution, keyboard entries are processed much slower because IOP0 is often busy testing Buffer Memory.

The following elements of the MBUF boot program are available to you:

- Parameters that control program execution and identify hardware
- Error information

† The MBUF test (dedicated) is based on the algorithm described by John Knaizuk, Jr. and C. R. P. Hartmann in IEEE TRANSACTIONS ON COMPUTER, April, 1977.

6.5.1 PARAMETERS

Table 6-7 lists the test parameters for MBUF and briefly describes the user options.

Table 6-7. MBUF parameters

Address	Default	Description
7	17	Identifies the processors to be tested. The default tests all processors. Selection is made by setting bits in the following way: *xxxxx*1 Processor 0 *xxxxx*2 Processor 1 *xxxxx*4 Processor 2 *xxxx*1*x* Processor 3
10	10	10 Sets Buffer Memory size in blocks of 1/8 million words (10=1 million words, 40=4 million words, and so on).
11	2	Sets the error channel reporting option. Selection is made by setting bits in the following way: *xxxxx*1 Report correctable errors *xxxxx*2 Report uncorrectable errors
12	1	Sets data checking. Selection is made by setting bits in the following way: 000000 Do not verify random read/write data. 000001 Verify data.
13	1	Sets the program stop condition
14	1001	Sets the maximum random block length to be used

6.5.2 ERROR INFORMATION

MBUF returns error information in two ways:

- Error messages
- Screen displays

MBUF error messages

Any errors encountered during a test run are recorded in the logfile on the right side of the screen.

Message	Meaning
CHAN 3 LM ERROR SEC/BK *xx* BYTE *x*	An error is detected during a random read or write.
CHAN 5 DIDNT INTERRUPT	An interrupt condition failed to occur.
CHAN *xx* DN AFTER DEADSTART CMD	The processor could not be deadstarted.
DATA ERROR DURING RANDOM READ XFER ADR BM *uuuuuu* 111 LM *xxxxxx* BLOCK LENGTH *xxxxxx* FAILING ADR BM *uuuuuu* 111 LM *xxxxxx* EXPECT *xxxxxx xxxxxx xxxxxx xxxxxx*† ACTUAL *xxxxxx xxxxxx xxxxxx xxxxxx*	An error was detected during a random read or write
DATA OR ADR ERROR ON 1 WORD XFER XFER ADR BM *uuuuuu* 111 LM *xxxxxx* BLOCK LENGTH *xxxxxx* EXPECT *xxxxxx xxxxxx xxxxxx xxxxxx* ACTUAL *xxxxxx xxxxxx xxxxxx xxxxxx*†	An error was detected while the memory was dedicated to a single processor.
LOST CHAN COMMUNICATION WITH A*x*	Accumulator channel status communication was lost with a processor.
UNEXPECTED INTERRUPT CHAN *xx*	An interrupt condition failed to occur.
ERRORLOG BM STATUS FIRST PARAMETER *xxxxxx* SECOND PARAMETER *xxxxxx* THIRD PARAMETER *xxxxxx*	Error log channel reports an error in Buffer Memory.
ERRORLOG LM STATUS A*x* FIRST PARAMETER *xxxxxx*	Error log channel reports an error in Local Memory.††

---

† The actual value represents random block transferred data, while the expected value is data obtained with a single-word read.

†† The error log channel is available on the CRAY-1 Models A, B, C and on the CRAY-1 S through serial number 20.

MBUF error display

An example of the MBUF error display is shown in figure 6-5.

```
LOGFILE.
**.MULTI PROCESSOR BUFFER MEMORY TEST
**.
**.MEMORY DEDICATED TO ONE PROCESSOR
A0.RUNNING
A0.DONE
A1.RUNNING
A1.DONE
A2.RUNNING
A2.DONE
A3.RUNNING
A3.DONE
**.MEMORY SHARED BY ALL PROCESSORS
**.DOING RANDOM READS AND WRITES
A0.RUNNING
A1.RUNNING
A2.RUNNING
A3.RUNNING
A1.DATA ERROR DURING RANDOM READ
A1.XFER ADR BM 000011 765 LM 010000
A1.BLOCK LENGTH 000324
A1.FAILING ADR BM 000011 766 LM 01000
A1.EXPECT  107324 051134 167102 000442
A1.ACTUAL 107324 051130 167102 000442
```

Figure 6-5. MBUF error display

6.6 BASIC CHECKOUT DISPLAY (BCD)

Basic checkout display (BCD) is a utility for displaying Local Memory and registers of IOP0 (see figure 6-6). It allows octal programs to be entered and executed.

BCD resides on its own deadstart tape and is deadstarted into IOP0 from the Peripheral Expander tape drive. To deadstart the program, follow the steps listed below.

1. Set the deadstart panel switches to their normal settings (see Appendix D.2.
2. Press the MASTER CLEAR button.
3. Press the DEADSTART button.
4. Press the SPACE key to enter the first character and begin execution.

								BCD	5/17/80
	LOCAL					REGISTERS			
10000.	026000	027001	074761	000000	0.	000012	177766	000000	000000
10004.	000000	000000	000000	000000	4.	000000	000000	000000	000000
10010.	000000	000000	000000	000000	10.	000000	000000	000000	000000
10014.	000000	000000	000000	000000	14.	000000	000000	000000	000000
10020.	000000	000000	000000	000000	20.	000000	000000	000000	000000
10024.	000000	000000	000000	000000	24.	000000	000000	000000	000000
10030.	000000	000000	000000	000000	30.	000000	000000	000000	000000
10034.	000000	000000	000000	000000	34.	000000	000000	000000	000000
10040.	000000	000000	000000	000000	40.	000000	000000	000000	000000
10044.	000000	000000	000000	000000	44.	000000	000000	000000	000000
10050.	000000	000000	000000	000000	50.	000000	000000	000000	000000
10054.	000000	000000	000000	000000	54.	000000	000000	000000	000000
10060.	000000	000000	000000	000000	60.	000000	000000	000000	000000
10064.	000000	000000	000000	000000	64.	000000	000000	000000	000000
10070.	000000	000000	000000	000000	70.	000000	000000	000000	000000
10074.	000000	000000	000000	000000	74.	000000	000000	000000	000000

Figure 6-6. BCD display

Programs must start at address 10000. Local Memory (10000-177777) and registers (0-700) are available. BCD does not use the exit stack. At the end of your instruction sequence, you must return control to the utility with a jump instruction. A jump to address 7600 or register 760 allows your program to execute once and stop. A jump to address 7610 or register 761 causes your program to loop until the ESC key is pressed.

Table 6-8 lists the commands available for running programs under the BCD utility and briefly describes the function of each.

Table 6-8. BCD commands

Command	Description
L *addr*	Displays Local Memory beginning at address *addr*
addr	Displays Local Memory beginning at address *addr*
L *addr value*	Sets Local Memory address *addr* to value *value*
addr value	Sets Local Memory address *addr* to value *value*

Table 6-8. BCD commands (continued)

Command	Description
LL *addr*	Displays the lower block of Local Memory at address *addr*
R *n*	Begins with register display with register *n*
R *n value*	Sets register *n* to value *value*
RU *n*	Displays the upper block of the register display with register *n*
RL *n*	Displays the lower block of the register display with register *n*
CL	Clears Local Memory
CR	Clears registers
WRITE *addr*	Writes 10000 parcels to tape from address *addr*
WRITE	Creates the BCD deadstart tape
REWIND	Rewinds the tape
READ *addr*	Reads a tape record to address *addr*

Table 6-9 lists the BCD function keys and gives a brief description of the operation of each.

Table 6-9. BCD function keys

Key	Description
RETURN	Executes a command
DEL	Erases the last character in the keyboard buffer
LINEFEED	Erases the keyboard buffer
TAB	Refreshes the entire display

Table 6-9. BCD function keys (continued)

Key	Description
SPACE	Starts the program at address 10000 (acts as the first character in the program)
ESC	Stops the execution of a running program

6.7 BASIC PROCESSOR INSTRUCTION TEST (BPX)

The basic processor instruction test (BPX) is an IOP0 diagnostic that resides on its own deadstart tape. To deadstart BPX into IOP0 with the Peripheral Expander tape drive, follow the steps listed below.

1. Set the deadstart panel switches to their normal settings.
2. Press the MASTER CLEAR button.
3. Press the DEADSTART button.

If the bell rings continuously on the IOP0 display, BPX is running. If the bell does not ring, scope the P register of IOP0 to determine the address that the processor is looping on. This address indicates an error stop which, along with the listing of BPX, should identify the failing instruction sequence. The P register can be examined by synchronizing on B32(2AT) and scoping F60-75(2AA).

6.8 DISK SYSTEM FLAW TEST (DKX)

The disk system flaw test (DKX) provides a system data integrity check and a disk surface analysis test. This subsection describes the control program that resides in IOP0 and the disk service program that resides in IOP1, IOP2, and IOP3.

The IOP0 control and disk driver communicate through the interprocessor channels. All data checks are done in IOP0. Buffer Memory is used for disk data storage and I/O command functions. The DKX control program reports error information, builds tables and sets flags that monitor control functions.

The display generated by the DKX diagnostic is shown in figure 6-7.

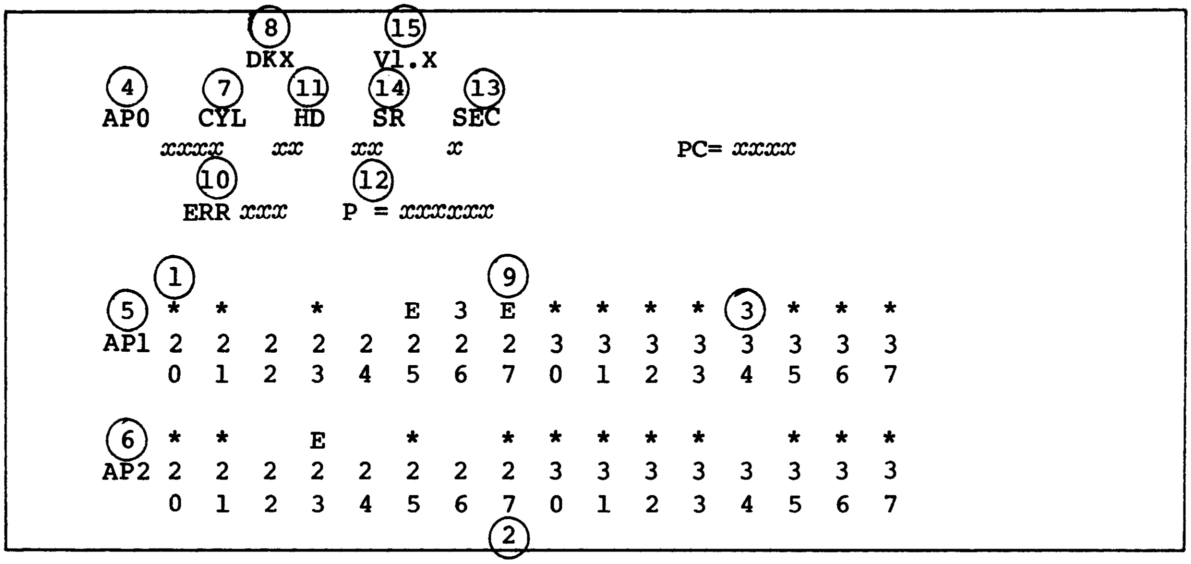


Figure 6-7. DKX error display

Cylinder, head, and sector display numbers are received from IOP *x* through Buffer Memory and may not reflect current values. The Disk Activity Table (DAT) is generated by the DKX program and contains current program values.

Descriptions for the keywords used in figure 6-7 are as follows.

Label	Comments
(1)	Disk Channel Busy flag set; represented by *.
(2)	Disk channels, a single digit from 0 through 7
(3)	Disk channels selected; represented by the digits 2 and 3.
(4)	IOP0 control program (AP0)
(5)	IOP1 selected (AP1)
(6)	IOP2 selected (AP2)
(7)	Cylinder (CYL)
(8)	Test name
(9)	Disk channel error (E)
(10)	Error ID number (ERR)
(11)	Head (HD)
(12)	Program halt address (P =)
(13)	Test section number (SEC)

(14) Sector (SR)

(15) Version number (V1.X)

Table 6-10 lists disk channel bit settings for data transfers. To activate any channel, set the bit associated with the channel equal to 1. The DKX program contains the following elements:

- Parameters
- Error messages

6.8.1 PARAMETERS

Test parameters can be set to control the following activities:

- Data transfer
- Test operation

Data transfer parameters

To activate any channel for data transfer, set the bit associated with the channel to 1. Table 6-10 lists the addresses for IOP1, IOP2, and IOP3 and the corresponding bits that must be set to activate the desired channel.

Table 6-10. Disk channel bit settings for data transfers

Address	Bit	Channel	Description
4001	0	20	IOP 1 disk channel selection
	1	21	
	2	22	
	3	23	
	4	24	
	5	25	
	6	26	
	7	27	
	8	30	
	9	31	
	10	32	
	11	33	
	12	34	
	13	35	
	14	36	
	15	37	

Table 6-10. Disk channel bit settings for data transfers (continued)

Address	Bit	Channel	Description
4002	0-15	20-37	IOP2 disk channel selection. The format for address 4002 is the same as address 4001.
4003	0-15	20-37	IOP3 disk channel selection. The format for address 4003 is the same as address 4001.

Test control parameters

DKX test control parameters set test flags, enable output display channels, set cylinder limits, and print flaw formats. Table 6-11 lists DKX control parameters and briefly describes the bit settings for the options available test. Test options are enabled using the method described above for data transfer channel selection.

Table 6-11. DKX test control parameters

Address	Section	Bit	Flag	Description
4004				Test section select flags
	0	00	1	Position and read/write cylinder 40.
	1	01	1	Select a random read/write cylinder.
	2	02	1	Select flaw mapping.
	3	10	1	Select read/write utility.
	4	20	1	Set user code.
4005				Section parameter flags
	0	00	1	Read data and log flaws.
	1	01	0	Write maintenance cylinder only.
	1	01	1	Random read/write to all cylinders.
	2	02	0	Select cylinder margins.
	2	02	1	Do not use cylinder margins.
4006		†		IOP1 output display channel
4007		†		IOP2 output display channel

† Low-order 6 bits

Table 6-11. DKX test control parameters (continued)

Address	Section	Bit	Flag	Description
4010		†		IOP3 output display channel
4011				Lower cylinder limit
4012				Upper cylinder limit
	0	00	1	Read data and log flaws.
	1	01	0	Write maintenance cylinder only.
	1	01	1	Randomly read/write all cylinders.
	2	02	0	Select cylinder margins.
	2	02	1	Do not use cylinder margins.
4013				Print flaw format selections
		00	0	Format selection 1.
		00	1	Format selection 2.
		15	1	Do not print flaws.
4014				Interrupt Disable flag for IOP0
		00	1	Disables Local Memory interrupts
		01	1	Disables error reporting channel
4015		00	1	Interrupt Disable flag for IOP*x*
4016				Log Disk Errors channel records channel 20 data errors for IOP1, IOP2, and IOP3 when address 4016=20. See table 6-13 for error information (IOP0).
4017		00	1	Command Look Ahead flag allows disk I/O commands (1-3) to be executed while IOP0 is checking data. Read and generate data commands are excluded.
4020				Pass Count flag (section 1 only)
			400	Default (400 passes)
			1	Minimum number of passes (1 pass)
			1777	Maximum number of passes (1777 passes)

† Low-order 6 bits

NOTE

Pressing the F key and the RETURN key terminates a DKX section. Pressing the SPACEBAR restarts the test.

6.8.2 ERROR MESSAGES

The DKX program returns the following kinds of error messages:

- Disk errors
- Channel initialization errors
- Miscellaneous errors
- Error information tables

Disk Error flags

Disk Error flags are set when disk identification errors occur. Table 6-12 lists and briefly describes messages generated when DKX Disk Error flags are set.

Table 6-12. DKX Disk Error flags

Flag	Description
171	Printer fault
203	Excessive parity errors; the error is flagged in IOP0.
204	ID error found because channel ID/head/sector/cylinder did not compare (match); the error is flagged in IOP0.
205	A data error is encountered; however, the Disk Error flag is not set. The error is flagged in IOP0.
206	40 Disk Error flags are set, but the data is checked as being good. A possible checkword circuit defect exists. The error is flagged in IOP0.

Table 6-12. DKX Disk Error flags (continued)

Flag	Description
220	Channel timeout
210	Function 6000† has a status of nonzero.
211	A cylinder status compare error is indicated.
212	A head status compare error is indicated.
213	A false error situation occurs when buffer data is equal to 1 and 6000 function† status is equal to 0.
214	Error status is bad (6000 function†).
215	Error (6000 function†) status is nonzero without the Error flag being set.
216	Disk channel is busy too long.
217	An extraneous disk interrupt has occurred.
560	A flaw table in IOP0 does not compare with flaws received from IOP*x*.
770	IOP0/IOP*x* timeout

† See the I/O Subsystem Reference Manual, CRI publication HR-0030.

Channel Initialization flags

Channel Initilization flags are set when channel initialization errors occur at deadstart or during data transfers between IOPs. Table 6-13 lists DKX Channel Initialization flags and briefly describes the function of each.

Table 6-13. DKX Channel Initialization flags

Flag	Description
61	Unable to communicate with IOP1 after deadstart (Buffer Memory flag)
62	Interprocessor output channel timeout; IOP0 transfer to IOP1.
63	Interprocessor input channel timeout; IOP0 transfer from IOP1.
64	Interprocessor Channel flag data error IOP1, IOP2, or IOP3.
101	Unable to communicate with IOP2 after deadstart (Buffer Memory flag)
102	Interprocessor output channel timeout; IOP0 transfer to IOP2.
103	Interprocessor input channel timeout; IOP0 transfer from IOP2.
121	Unable to communicate with IOP3 after deadstart (Buffer Memory flag)
122	Interprocessor output channel timeout; IOP0 transfer to IOP3.
123	Interprocessor input channel timeout; IOP0 transfer from 1OP3.

Miscellaneous DKX error flags

Miscellaneous error flags are set if data transfers are interrupted or if data transfers exceed time limits. Table 6-14 lists miscellaneous error flags and briefly describes the function of each.

Table 6-14. Miscellaneous DKX error flags

Flag	Description
31	Local Memory parity error; parity error halts tests in IOP0 through IOP3.
51	Buffer Memory Error flag; Buffer Memory error halts tests in IOP0 through IOP3.

Table 6-14. Miscellaneous DKX error flags

Flag	Description
52	Buffer Memory Error flag; extraneous buffer interrupt halts test in IOP0 through IOP3.
66	Interprocessor Channel Error flag (input); illegal function code causes a test halt in IOP1, IOP2, or IOP3.
72	Interprocessor Channel Error flag (output); function response to IOP1 is too long.
112	Interprocessor Channel Error flag (output); function response to IOP2 is too long.
132	Interprocessor Channel Error flag (output); function response to IOP3 is too long.
161	Error Report channel (IOP0 only); IOP0 parity error halts test in IOP0 through IOP3.
162	Error Report channel (IOP0 only); IOP2 parity error halts tests in IOP0 through IOP3.
163	Error Report channel (IOP0 only); IOP3 parity error halts tests in IOP0 through IOP3.
164	Error Report channel (IOP0 only); Buffer Memory error.
165	Error Report channel (IOP0 only); erroneous error.

The operand registers contain the following error information:

Register	Contents
EL	Local Memory error (byte/section/bank)
ES†	Error status
EA†	First parameter
EB†	Second parameter
EC†	Third parameter

† This register is used by the error reporting channel.

Error information tables

Error information tables are generated by the DKX diagnostic test. Table 6-15 lists and briefly describes these error tables. For more information about any of the tables described in table 6-15, see DKX in the I/O Subsystem (IOS) Diagnostic Ready Reference Guide, CRI publication HQ-1007.

Table 6-15. DKX error information tables

Table	Description
IOP0 Disk Error flags	Contains disk error information for channels 20_8 through 37_8 of IOP1, IOP2, and IOP3
Data Error Buffer (DEB)	Contains disk error information for IOP1, IOP2, IOP2, and IOP3 starting at address 22000
Data error history (DEHB)	Contains the disk data errors for the Buffer channel selected at address 4016
ID error history Buffer (IEB)	Contains disk ID header errors
Disk activity buffer (DAX)	Sets aside an area for IOP1, IOP2, and IOP3
Disk Activity Table (DAT)	Contains current error information on the following: I/O function, cylinder, head, sector status, and error flags
Disk I/O function stack (DSKA)	Contains the last 10 I/O functions executed at address 4400
Buffer Memory disk sector map	Creates a data storage area in Buffer Memory for channels 20_8 through 37_8 of IOP1, IOP2, and IOP3
Flaw Tables	Creates an area in Local Memory for flaw information

6.9 SAVE A ZERO TEST (SAZ)

The save a zero (SAZ) test resides on its own deadstart tape. It sends the image in IOP0 to an area in Buffer Memory where it can be subsequently dumped to the printer with the DUMP boot. The SAZ tape should be deadstarted once (no display appears). To print a listing of the IOP0 image on the line printer, do the following:

1. Mount and deadstart the IOP0 deadstart tape.

2. Select the DUMP boot.

DISK AID ROUTINES 7

This section describes the following disk aid routines that are available to check data transfers to and from disks:

- Disk aid interpreter (DSK)
- Multipe disk-aid interpreter (DSKM)

7.1 DISK AID INTERPRETER (DSK)

The disk aid (DSK) interpreter is a microcode interpreter for IOP/DCU-4. DSK is a utility that facilitates data and function transfers to and from disk storage units (DSUs). DSK builds functions from microcode instructions, sets up I/O sequences, checks for time-outs and error flags, and allows data generation and checking. DSK allows you to interact with disk drives and the controller without being concerned with the detailed programming of an I/O sequence.

DSK uses the PTA monitor (see Appendix B.2) and resides in the lower 4000 parcels of I/O Processor (IOP) Local Memory. Microcode instructions reside in a table that starts at address 4000 and ends at address 7557. When deadstarted, DSK executes micro instructions beginning with the instruction at address 4000. Each micro instruction is read, interpreted into APML, and executed.

Important elements of DSK include the following:

- Microcode instruction set
- Counters and flags

7.1.1 MICROCODE INSTRUCTIONS

Microcode instructions† are four parcels in length and must start at word boundaries. The first parcel holds the instruction and the three

† COMPARE DATA (11) is the only microcode instruction that is not four parcels in length; it is eight.

remaining parcels contain the required parameters. Microcode instructions are divided into the four basic classes. Use instruction classifications in problem identification and resolution.

Class	Description
Functions (F)	Actual function to the DSU or controller
Program flow (P)	Jumps and conditional jumps
Data manipulation (D)	Generating, moving, and checking data
Operations (O)	Arithmetic or logical operations

NOTE

Unlike the Cray CPU version of disk aid, line numbers are not biased. All references to memory locations or line numbers are real addresses.

With the exception of functions and real addressing, the IOS disk aid version is basically the same as the Cray CPU version.

Table 7-1 lists the microcode instruction set and tells how each command should be entered. The basic class for each instruction is shown in parenthesis following each command (F for function, P for program flow, D for data manipulation, and O for operation).

Table 7-1. Microcode instruction parcel format

Command	Parcel 0	Parcel 1	Parcel 2	Parcel 3
PASS (F)	0	None	None	None
SELECT MODE/ STATUS (F)	1	Function address	Response (0=Stop)	Error line
READ (F)	2	0	Buffer address	Error line
WRITE (F)	3	0	Buffer address	Error line

Table 7-1. Microcode instruction parcel format (continued)

Command	Parcel 0	Parcel 1	Parcel 2	Parcel 3
SELECT HEAD (F)	4	0	None	Error line
SELECT CYL/READ ID (F)	5	0	None	Error line
SELECT CYL/ABORT (F)	5	1	None	Error line
SELECT CYLINDER (F)	5	2	None	Error line
READ ID (F)	5 5	3 3	None	Error line
ENTER STATUS (F)	7	0	Data	None
ENTER ADDRESS (F)	7	1	Data	None
READ STATUS (F)	7	2	Response address	None
READ ADDRESS (F)	7	3	Response address	None
GENERATE DATA (D)	10	Buffer address	0	Data
GENERATE DATA INCREMENT PARCEL (D)	10	Buffer address	$2^{15}=1$	Data

Table 7-1. Microcode instruction parcel format (continued)

Command	Parcel 0	Parcel 1	Parcel 2	Parcel 3
GENERATE DATA FORMAT (D)	10	Buffer address	2^{14}=1	Data
GENERATE DATA LINE (D)	10	Buffer address	2^{12}=1	Line address
COMPARE DATA (D 2 words)	11	Check data (1st error)	Read data bits in error error †	Error line parcel in
GO TO LINE (P)	12	Line number		
LOOP TO LINE (P)	13	Line number	Pass count	0 (used by program)
CLEAR/SET FNT (D)	14	1=Set 0=Clear	None	None
PAUSE (P)	15	None	None	None
GENERATE CHECK WC (D)	16	Parcel count	None	None
SET LOCATION (D)	17 ††	Parcel address	Data	None
IF= (P)	20 ††	Parcel address	Data	Match line (0=Stop)
IF= (P)	21 ††	Parcel address	Data	Match line

† If bit 2^{15} of parcel 3=1, the compare is a short compare. Only the first word of the check data is used in the compare.

†† If bit 2^{15} is set in parcel 1, parcel 2 becomes the parcel address for the second operand.

Table 7-1. Microcode instruction parcel format (continued)

Command	Parcel 0	Parcel 1	Parcel 2	Parcel 3
IF (P)	22 †	Parcel address	Data	Match line
IF (P)	23 †	Parcel address	Data	Match line
IF (P)	24 †	Parcel address	Data	Match line
IF (P)	25 †	Parcel address	Data	Match line
IF POSITIVE (P)	26	Parcel address	None	Match line
IF NEGATIVE (P)	27	Parcel address	None	Match line
SET HEAD (D)	30	Head	Unit	None
SET SECTOR (D)	31	Sector	Unit	None
SET UNIT (D)	32	Unit	None	None
SET CYLINDER (D)	33	Cylinder	Unit	None
AND LOCATION (O)	34 †	Parcel address	Data	Destination parcel address
ADD LOCATION (O)	35 †	Parcel address	Data	Destination parcel address
MASTER CLEAR (F)	36	Unit	None	None

† If bit 2^{15} is set in parcel 1, parcel 2 becomes the parcel address for the second operand.

Table 7-1. Microcode instruction parcel format (continued)

Command	Parcel 0	Parcel 1	Parcel 2	Parcel 3
END (P)	37	None	None	None
EXCLUSIVE OR LOCATION (O)	40 †	Parcel address	Data	Destination parcel address
SHIFT RIGHT (O)	41	Parcel address	Shift count	Destination parcel address
SHIFT LEFT (O)	42	Parcel address	Shift count	Destination parcel address
GENERATE RANDOM ADDRESS (D)	43	0	0	0
FOR DUAL DENSITY DSU (D)	43	1	0	0
GENERATE RANDOM NUMBER (D)	43	2	Limit	Destination parcel address

† If bit 2^{15} is set in parcel 1, parcel 2 becomes the parcel address for the second operand.

7.1.2 DSK COUNTERS AND FLAGS

The addresses for the counters and flags used by DSK are listed below.

Address	Description
7566	Dead mode - ignore busy
7567	Present line of execution
7570	Format flag

7571	Unit bias
7572	Word count
7573	ID received
7574	Function delay
7575	Response delay
7576	Data delay
7577	I/O Error flag. If one of the following values is found in address 7577, the corresponding condition exists:

000000	Transfer is good
000001	Error on function
000002	Time out
000005	ID error
100001	Error on second function or status
100002	Error on second function or status

7600 Drive Table

Parcel 0	Parcel 1	Parcel 2	Parcel 3
Channel number	Head	Sector	Cylinder

7700	Last Access Table
7720	Delay Table

7.2 MULTIPLE DISK-AID INTERPRETER (DSKM)

The multiple disk-aid interpreter (DSKM) is an IOP-resident disk driver capable of driving up to 16 DD-19s or DD-29s simultaneously. DSKM is microcode driven, resides in addresses 0 through 4000 and interprets microcode instructions that reside in the execution buffer; addresses 4000 through 7474.

Important elements of DSKM include the following:

- Descriptor Table
- Buffers
- Summary Table
- Error codes

Table 7-2 contains the addresses for the tables (Summary, Descriptor, and I/O Buffer) generated by the DSKM program.

Table 7-2. DSKM table addresses

Unit	Channel	Summary	Descriptor	I/O buffer
0	20	7600	10000	20000
1	21	7604	10040	24000
2	22	7610	10100	30000
3	23	7614	10140	34000
4	24	7620	10200	40000
5	25	7624	10240	44000
6	26	7630	10300	50000
7	27	7634	10340	54000
10	30	7640	10400	60000
11	31	7644	10440	64000
12	32	7650	10500	70000
13	33	7654	10540	74000
14	34	7660	10600	100000
15	35	7664	10640	104000
16	36	7670	10700	110000
17	37	7674	10740	114000

7.2.1 DESCRIPTOR TABLE

Each disk storage unit (DSU) has its own 40-parcel Descriptor Table that contains important information about the unit's status, and the locations reserved for microcode or user data. Table 7-3 contains addresses for the Descriptor Tables for each unit.

Table 7-3. DSKM Descriptor Table addresses

Address	Parcel 0	Parcel 1	Parcel 2	Parcel 3
10000	Channel	Head	Sector	Cylinder
10004	Disk type	Word count	Expected ID	Actual ID
10010	Pass count	Error code	Response wait	Line number
10014	Loop count 0	Loop count 1	Loop count 2	Loop count 3

Table 7-3. DSKM Descriptor Table addresses (continued)

Address	Parcel 0	Parcel 1	Parcel 2	Parcel 3
10020 †	T0,0	T0,1	T0,2	T0,3
10024 †	T1,0	T1,1	T1,2	T1,3
10030 †	T2,0	T2,1	T2,2	T2,3
10034 †	T3,0	T3,1	T3,2	T3,3

† Scratch registers; T0,0 is register T0, parcel 0; T0,1 is register 0, parcel 1 and so on.

7.2.2 WRITE/READ BUFFERS

Each unit has its own 4000-parcel I/O buffer for reads and writes. Table 7-2 lists the respective buffer addresses for each DSU and channel.

7.2.3 SUMMARY TABLE

DSKM has a Summary Table with a 4-parcel entry for each unit. The Summary Table allows you to monitor the pass and error counts of each drive and contains the address of the Descriptor Table. The Descriptor Table addresses (DTA) appear only if the drive is selected.

The Summary Table starts at address 7600 (table 7-2) and has the following format:

Address	Parcel 1	Parcel 2	Parcel 3	Parcel 4
7600	Channel	DTA	Pass count	Error code
7604	Channel	DTA	Pass count	Error code
.	.	.	.	.
.	.	.	.	.
7634	Channel	DTA	Pass count	Error code

7.2.4 ERROR CODES

If an error occurs during the DSKM test, error codes appear in parcel 3 of the erring unit's Summary Table entry. DSKM error codes are as follows:

Code	Description
1	Function error
2	Read error
3	Write error
4	Response error
5	ID error
6	Microcode violation (software)
11	Compare error

In addition to the error codes listed above, microcode can pass its own user-defined error codes through parcel 1 of the exit instruction. See the appropriate microcode listing for the definitions of user-defined error codes. More error information is made available by displaying the appropriate Descriptor Table. See table 7-2 for the addresses of the Descriptor Table and subsection 7.3.1 for the Descriptor Table format. The Descriptor Table contains the following error information:

Error data	Description
Line number	Last microcode line executed
Head	
Sector	Last write or read address=1 sector
Cylinder	
ID actual	ID read on last position function
ID expected	ID expected (valid only on ID errors)
Last status	Status taken after the error occurred

Table 7-4 lists the microcode instruction set and indicates the values that should be entered in each parcel of the 4-parcel package.

Table 7-4. DSKM microcode instruction set

Macro	Description	0	1	2	3
PA	Pass	0	0	0	Error line
RL	Release unit	1	0	0	Error line
RS	Reserve unit	1	1000	0	Error line
CF	Clear fault	1	2000	0	Error line
RTZ	Return to zero	1	3000	0	Error line
MR	Set margin	1	4000	0	Error line

Table 7-4. DSKM microcode instruction set (continued)

Macro	Description	0	1	2	3
STS	Status sector	1	5000	Address	Error line
STEF	Status error flags	1	6000	Address	Error line
STC	Status cylinder	1	7000	Address	Error line
STH	Status head	1	7001	Address	Error line
STM	Status margin	1	7002	Address	Error line
STI	Status interlock	1	7003	Address	Error line
RD	Read	2	Buffer	0	Error line
WT	Write	3	Buffer	0	Error line
HS	Head select	4	0	0	0
PO	Position	5	0	0	Error line
POA	Position/abort	5	1	0	Error line
CSF		6	0	0	0
WST	Write status register	7	0	0	0
WAD	Write address register	7	1	0	0
RST	Read status register	7	2	Address	0
RAD	Read address register	7	3	Address	0
GD	Generate data	10	Destination	0	Data
GDL	Generate data line	10	Destination	1	Line address
GDL@	Generate data line at	10	Destination	2	Address

Table 7-4. DSKM microcode instruction set (continued)

Macro	Description	0	1	2	3
GDF	Generate data format	10	Destination	4	0
GDA	Generate data address	10	Destination	10	Start address
CD	Compare data	11	Buffer A	Buffer B	Error line
CDS	Compare data w/line	11	Buffer	Line address	Error line
J	Jump to line	12	Line	0	0
DL	Loop to line	13	Line	Count	Pass address
SM	Set/clear special mode	14	$ MODE	0	0
WC	Set word count	16	Count	0	0
SP	Set parcel	17	Parcel	Data	0
SPP	Set parcel to parcel	17†	1 parcel	1 parcel	0
SPP@	Set parcel PRC at	17††	1 parcel	Parcel address	0
EQ	Equal to	20	Parcel	Data	Match line
EQP	Equal to parcel	20†	1 parcel	Parcel	Match line
EQP@	Equal to parcel	20††	1 parcel	Parcel address	Match line
NE	Not =	21	Parcel	Data	Match line

† If bit 2^{15} is set in parcel 1, parcel 2 contains the addresses of the second operand for the operation.

†† If bit 2^{15} of parcel 2 is set, parcel 2 contains the addresses of the addresses of the second operand for the operation.

Table 7-4. DSKM microcode instruction set (continued)

Macro	Description	0	1	2	3
NEP	Not = to parcel	21†	1 parcel	Parcel	Match line
NEP@	Not = to parcel at	21††	1 parcel	parcel address	Match line
LT	Less than	22	Parcel	Data	Match line
LTP	Less than parcel	21†	1 Parcel	Parcel	Match line
LTP@	Less than parcel at	22††	1 Parcel	Parcel address	Match line
GT	Greater than	23	Parcel	Data	Match line
GTP	Greater than parcel	23†	1 Parcel	Parcel	Match line
LTP@	Less than parcel at	23††	1 Parcel	Parcel address	Match line
LE	Less than or equal	24	Parcel	Data	Match line
LEP	Less than or equal parcel	24†	1 parcel	Parcel	Match line
LEP@	Less than or equal parcel at	24††	1 parcel	1 parcel	Match line
GEP@	Greater or equal parcel at	24††	1 parcel	1 parcel	Match line
GE	Greater than or equal parcel	25	Parcel	Data	Match line

† If bit 2^{15} is set in parcel 1, parcel 2 contains the addresses of the second operand for the operation.

†† If bit 2^{15} of parcel 2 is set, parcel 2 contains the addresses of the addresses of the second operand for the operation.

Table 7-4. DSKM microcode instruction set (continued)

Macro	Description	0	1	2	3
GEP	Greater or equal parcel	25†	1 parcel	Parcel	Match line
PS	If positive	26	Parcel	0	Match line
NG	If negative	27	Parcel	0	Match line
SH	Set head	30	Data	0	0
SS	Set sector	31	Data	0	0
SC	Set cylinder	32	Data	0	0
An	And	34	Parcel	Data	Destination parcel
ANP	And to parcel	34†	1 parcel	Parcel	Destination parcel
AD	Add	35	Parcel	Data	Destination parcel
ADP	Add to parcel	35†	1 parcel	Parcel	Destination parcel
MC	Master clear	36	Parcel	Data	Destination parcel
Exit	EX	37	Error code		
XO	Exclusive or	40	Parcel	Data	Destination parcel
XOP	Exclusive or parcel	40†	1 parcel	Data	Destination parcel
SHR	Shift right	41	Parcel	Out	Destination parcel

† If bit 2^{15} is set in parcel 1, parcel 2 contains the addresses of the second operand for the operation.

†† If bit 2^{15} of parcel 2 is set, parcel 2 contains the addresses of the addresses of the second operand for the operation.

Table 7-4. DSKM microcode instruction set (continued)

Macro	Description	0	1	2	3
SHL	Shift left	42	Parcel	Out	Destination parcel
GRN	Gen random	43	1	Limit	Destination
GRA	Gen random address numnber	43	0	0	0 parcel
GRNP	Generate random number parcel	43	2	Limit parcel	Destination parcel
GFC	Generate fire code	44	Buffer	Destination	0
FLI	Flaw Table initialize	46	0	0	0
FLR	Record flaw	46	1	0	0
FLS	Flaw Table sort	46	2	0	0

The modes for read or write operations are:

Mode	Description
0	Clear read/write mode
1	Set mode to write format
2	Set mode to read or code

If bit 2^{15} is set in parcel 1, parcel 2 contains the second operand for the operation. @ signifies that parcel 2 contains the addresses of the address containing the operand.

If address equals 0, it is interpreted as the units write/read buffer. If an address is less than 40, it is added to the unit's Descriptor Table address. If address is greater than or equal to 40, it is interpreted as the real address.

ONLINE I/O SUBSYSTEM TESTS 8

Two types of online I/O Subsystem (IOS) diagnostic tests are available. The first type is made up of diagnostics that run under the control of the diagnostic online monitor (DOM) and include the following: BMOL, CPOL, F80M, LPT, and MAGR. The second type are called system diagnostics and include: CHNTST, HSPTEST, MOSTEST, XDK, XMT, and XPR.

8.1 DIAGNOSTIC ONLINE MONITOR (DOM)

The diagnostic online monitor (DOM) runs as an overlay under the control of the IOS Kernel. Therefore, the Cray CPU does not have to be operational to run programs under the control of the online monitor. DOM can be loaded into each I/O Processor (IOP) by entering the name on the console attached to that IOP. To begin the process, enter:

```
DOM
```

When DOM has been loaded, the following message is displayed:

```
DIAGNOSTIC ONLINE MONITOR ACTIVE
LOAD WHAT?
```

To load a diagnostic overlay, enter the diagnostic name (BMOL, CPOL, F80M, LPT, or MAGR). For example, entering the following command loads BMOL:

```
BMOL
```

DOM verifies the existence of a diagnostic overlay of that name. It also verifies that the diagnostic can be run in the IOP where the name was entered. If the overlay program does not exist, or if it does exist but cannot be run in that particular IOP, DOM displays the following message:

```
*INVALID OVERLAY NAME*
```

DOM restarts the process and asks what to load.

If the diagnostic overlay does exist, and can be run in that IOP, DOM attempts to read in a Parameter Table. If the DOM's Parameter Table exists, DOM checks to see if a text display of unique keywords exists in the diagnostic. If the keywords exist, DOM reads in and displays this text as shown below:

OPTIONS ARE:

ICHN=*input channel*
OCHN=*output channel*
.
.
DEV=*device*

DOM asks for parameters by displaying the following message:

ENTER PARAMETERS?

Parameters can be entered one at a time, or several can be entered on the same line if they are separated by commas. Each entry is terminated by pressing the RETURN key. A parameter table of standard DOM keywords is described in table 8-1.

Table 8-1. DOM keywords†

Keyword	Function	Comment
ICHN	Input channel	Used for interfaces
OCHN	Output channel	Used for interfaces
ITYPE	Interface type	
CMODE	Interface mode	Can be master, slave, or loop back
CHAN	Channel number	Used for peripherals
DEV	Device address	Used for peripherals
SECT	Section selects	To change default sections selected

† Values for the DOM keywords are entered in octal.

The program stop condition controls the operation of the programs running under DOM. Changing the stop condition, changes the information that is generated by the test. DOM stop conditions can be one condition or a combination of conditions. DOM stop conditions are listed in table 8-2.

Table 8-2. Stop conditions for DOM

Command	Function
SE	Stop on error
SSC	Stop at the end of the subcondition
SC	Stop at the end of the condition
SSS	Stop at the end of the subsection
SS	Stop at the end of the section
ST	Stop at the end of the test

The stop conditions are turned on or off as shown by the examples listed below:

ON=ST or ON=ST,SS

OFF=ST or OFF=ST,SS

The program repeat condition controls the operation of programs running under DOM. Changing the repeat condition, changes the information that is generated by the test. The repeat condition can be one condition or a combination of conditions. DOM repeat conditions are listed in table 8-3.

Table 8-3. DOM repeat conditions

Command	Function
CONT	Continue flag
SCOP	Scope Loop flag
LE	Loop on error
RSC	Repeat subcondition
RC	Repeat condition
RSS	Repeat subsection
RS	Repeat section
RT	Repeat test

The repeat conditions can be turned on or off as shown by the following examples:

ON=LE or ON=LE,CONT

OFF=LE or OFF=LE,CONT

Further parameters unique to a particular diagnostic can be in each diagnostic. DOM provides a display description of each keyword.

DOM searches for ON= or OFF= parameters in its own table only. Any other entry causes a search of a parameter table in the diagnostic, followed by a search of DOM's Parameter Table. If a keyword cannot be found, the following message is displayed:

INVALID PARAMETER
ENTER PARAMETERS?

When all of the parameters have been entered, start execution by entering:

GO

To stop execution, enter:

STOP DOMTST

The following message is displayed after STOP DOMTST is entered:

xxxx TERMINATED
ENTER PARAMETERS

To restart a diagnostic with different parameters, after a stop on error or an end of section test, do the following:

1. Type RESET.
2. Press the RETURN key.
3. Enter the new parameters.

To terminate a diagnostic after a stop on error or STOP DOMTST, enter:

TERM

8.1.1 ONLINE BLOCK MUX/STC TAPE TEST (BMOL)

The online block MUX/STC tape test (BMOL) is called by the diagnostic online monitor (DOM) and its options are displayed on the console. The test returns error information stored in error buffers and generates error messages on the screen. BMOL acquires a block Mux channel, a control unit, and all the drives attached to that unit.

NOTE

No mount message is displayed. The program assumes that a tape is to be mounted and waits until the tape is mounted.

The list of options includes the following:

Option	Description
CHAN	Block MUX channel to be tested.
DEV	Logical device address (device ordinal) of an STC tape drive.
SECT	Section select bits where, 1=BM register test 2=Fixed block read/write (default) 3=Request in test 4=LADDER test

BMOL consists of four test sections. The default runs the test in section 2 only.

Section 1

Section 1 loads and reads back all accessible block Mux registers using the following patterns:

- Pattern 1=000000
- Pattern 2=177777
- Pattern 3=125252
- Pattern 4=052525

If an error occurs, the pattern expected and the pattern received are displayed on the console along with the name of the failing register.

Section 2

Section 2 runs with interrupts locked out in the sequence listed below:

1. Writes using command chaining

2. Writes two tapemarks and rewinds

3. Reads what was just written using read commands, without command chaining, and verifies the data

Section 3

Section 2 must have been run before running section 3. Section 3 runs in the sequence listed below:

1. Tests requests made in processing by causing busy conditions using forward space file and backward space file commands

2. Issues a command to an illegal divide address (HEX FF) and verifies that the control unit rejects it

3. Rewinds the tape and reads to the end of file (two tape marks)

4. Verifies that the block number and word number in the first and second block read are correct

Section 4

Section 4 is a ladder test that runs in the sequence shown below:

1. Writes out 1-byte records and increments by 1 until a size of 256 is reached

2. Increments by 64 until the maximum size (octal 1000000) is reached

3. Writes two tapemarks and rewinds

4. Reads and verifies data

Error information

Error information is returned in the form of error buffers and messages.

Buffer configuration - Buffers for sections 2, 3, and 4 consist of four parcels that contain the following data:

Parcel	Data
1	Buffer address
2	Block count

Parcel	Data
3	Block size in characters
4	Number of the 4-parcel group

Error messages - All errors are displayed on the console in English. When problems occur, sense bytes are displayed in hexadecimal. The hexadecimal display helps communication between system test and check out (STCO) personnel. BMOL error messages are described below.

Message	Description
ADDRESS REGISTER ERROR EXP=*xxxxxx* ACT=*yyyyyy*	Section 1 detected an error loading the expected value (*xxxxxx*) into the address register. The pattern shown as actual data (*yyyyyy*) was the pattern returned when the register was read back.
DATA ADDRESS REGISTER ERROR EXP=*xxxxxx* ACT=*yyyyyy*	Section 1 detected an error loading the expected value (*xxxxxx*) into the data address register. The pattern shown as actual (*yyyyyy*) was the pattern returned when the register was read back.
DONE TIMEOUT	The Done flag was not set before a timeout occurred.
INTERFACE ERRORS DATA ERROR EXP=*xxxxxx* ACT=*yyyyyy*	The program detected an interface error. An error occurred between the data read (*xxxxxx*) and the data written (*yyyyyy*).
NO BUFFER SPACE	The program was unable to acquire a buffer from the system.
OPEN ERRORS	The program was unable to acquire a block Mux channel, control unit, and tape drives from the system.
POSITION ERROR	Section 3 detected a positioning error.
STATUS ERROR	The following error information is returned when a status error occurs.

STATUS=*xx*

COMMAND=*yy*	BLOCK SIZE = 000000		COUNT = 000000
SENSE = 01 *ss*	02 *ss*	03 *ss*	04 *ss*
= 05 *ss*	06 *ss*	07 *ss*	08 *ss*
= 09 *ss*	10 *ss*	11 *ss*	12 *ss*
= 13 *ss*	14 *ss*	15 *ss*	16 *ss*
= 17 *ss*	18 *ss*	19 *ss*	20 *ss*
= 21 *ss*	22 *ss*	23 *ss*	24 *ss*

Where *xx* is equal to STC status byte in hexadecimal, *yy* the command issued, and *ss* the sense byte.

8.1.2 BLOCK MUX TAPE CONFIDENCE TEST (CPOL)

The block Mux tape confidence Test (CPOL) is called by the diagnostic online monitor (DOM) and its option is displayed on the console as follows:

DEV=*lda*

lda is the logical device address (device ordinal) of an STC tape drive.

The block Mux tape confidence (CPOL) has one test section. It uses CPW lists to build channel programs that are executed by the block MUX software in the following sequence:

1. Writes fixed blocks

2. Reads backward

3. Reads forward

If stop on error is set, CPOL stops and waits for input when an error is encountered. To start the program over, enter:

GO

The only required input parameter is DEV=, the device ordinal, which gives access to a tape drive depending on the path through the system.

NOTE

No mount message is displayed. The program assumes a tape is to be mounted and waits until it is.

Errors are reported in English. Status and sense are reported in hexadecimal. CPOL returns the following error messages:

Message	Description
INTERFACE ERRORS	The program detected an interface error.
NO BUFFER SPACE	The program was unable to acquire a buffer from the system.
OPEN ERRORS	The program was unable to acquire a block Mux channel, control unit, and tape drives from the system.
STATUS ERROR	The following error information is returned when a status error occurs.
STATUS=*xx*	Where *xx* is equal to STC status byte in hexadecimal, *yy* the command issued, and *ss* the sense byte.

COMMAND=*yy*	BLOCK SIZE = 000000		COUNT = 000000
SENSE = 01 *ss*	02 *ss*	03 *ss*	04 *ss*
= 05 *ss*	06 *ss*	07 *ss*	08 *ss*
= 09 *ss*	10 *ss*	11 *ss*	12 *ss*
= 13 *ss*	14 *ss*	15 *ss*	16 *ss*
= 17 *ss*	18 *ss*	19 *ss*	20 *ss*
= 21 *ss*	22 *ss*	23 *ss*	24 *ss*

8.1.3 F80M

F80M is a formatter and diagnostic for the AMPEX 80 Mbyte disk attached to the IOP0 Peripheral Expander chassis. For a detailed explanation of F80M, see the IOS Software Internal Reference Manual, CRI publication SM-0046.

8.1.4 LINE PRINTER TEST (LPT)

The line printer test (LPT) runs in IOP0 and tests the Gould printer on the Peripheral Expander. LPT is called by DOM and its options are displayed on the console as follows:

Mode	Description
1	Print high-speed graphics
2	Print low-speed graphics
4	Print alpha characters

When LPT is entered, the system displays the following message:

ENTER PARAMETERS?

If a specific mode is not entered, all modes are run. The diagnostic runs until the following command is entered at the console:

STOP DOMTST

When STOP DOMTST is entered, the system displays the following messages:

LPT TERMINATED
ENTER PARAMETERS?

To return control to the Kernel, enter:

TERM

8.1.5 MAGNETIC TAPE RELIABILITY TEST (MAGR)

The magnetic tape reliability test (MAGR) runs in IOP0 and writes a random number of records containing a random word count in the following sequence:

1. Rewinds the tape
2. Writes data
   a. Gets record count
   b. Gets record word count (2 through 7777)
   c. Gets data
   d. Writes data to tape
   e. Goes to step b record count times
3. Backspaces record count records

4. Reads data

   a. Resets random number generator
   b. Gets word count
   c. Reads data from tape
   d. Verifies data
   e. Goes to the beginning of step 4 record count times

5. Goes to step 2

MAGR returns the following error messages when errors are encountered:

Message	Description
DATA COMPARE ERROR EXP=*xxxxxx* ACT=*yyyyyy*	The data written (*xxxxxx*) was not the same as the data read (*yyyyyy*).
OPEN ERRORS	MAGR was unable to acquire the Peripheral Expander tape from the system.
BUFFER NOT AVAILABLE	MAGR was unable to obtain a buffer from the system.
READ PARITY ERRORS	A read parity error was detected.

8.2 SYSTEM TESTS

System tests are released as part of the system software. Enter the name of the desired test (CHNTST, HSPTEST, MOSTEST, XDK, XMT, or XPR) at the MIOP Kernel console, and press the RETURN key to begin execution. All system tests run as overlays under control of the Kernel.

When the test name is entered at the Kernel console, the Kernel checks the Overlay Table for the test name. If the test exists, the Kernel places the diagnostic into execution.

The STOP command terminates MOSTEST, HSPTEST, and CHNTST. Enter the STOP command at the Kernel console of each IOP in which the diagnostic is active as follows:

STOP *test*

test is the name of the online diagnostic test.

To terminate XDK, XMT, and XPR, enter one of the following ABORT commands at the IOP0 Kernel console:

ABORT DK0 Terminates XDK
ABORT MT0 Terminates XMT
ABORT PR0 Terminates XPR

The test being halted displays an abort message.

8.2.1 CHNTST

The CHNTST diagnostic is a channel loop-back test. CHNTST verifies reliable data transfer on the Cray 6 Mbyte (low-speed asynchronous) channel.

Before running CHNTST, connect the input and output cables of the channel pair being tested with the one foot cable assembly (part number 2203505) that is specially made for this purpose.

The station error display lists any errors encountered. Enter the ERROR command at the station console to obtain the display. The error display indicates an input or output channel error, status, data expected, data received, and an input or output channel time out.

NOTE

CHNTST cannot be run concurrently with other software.

To load the diagnostic overlay, enter the CHNTST command at the IOP0 Kernel console as follows:

```
CHNTST
```

The diagnostic runs until an error is encountered or the STOP CHNTST command is entered at the IOP0 Kernel console.

8.2.2 HSPTEST

HSPTEST creates a high level of activity on all of the 100 Mbyte (high-speed) channels that are configured. HSPTEST tests the Cray mainframe with writes to and reads from each 512-word block of Central Memory through the 100 Mbyte channel. The block sizes vary from 1 to 512 words, and varying data patterns are used.

The station error display lists any errors encountered. Enter the ERROR command at the station console to obtain the error display. The error display gives an address, the data expected, and the data received. HSPTEST displays a PASS COMPLETE message each time all of Central Memory has been tested.

NOTE

HSPTEST cannot be run concurrently with other software.

To load the diagnostic overlay, enter the HSPTEST command at the IOP0 Kernel console as follows:

```
HSPTEST
```

When HSPTEST is loaded, the diagnostic displays the following message:

```
THIS TEST WRITES OVER CPU MEMORY.
DO YOU REALLY WANT TO RUN IT?
```

To run HSPTEST, enter YES.

The diagnostic runs until the STOP HSPTEST command is entered at the console of each IOP in which HSPTEST is running.

8.2.3 MOSTEST

The MOSTEST diagnostic generates a high level of Buffer Memory I/O on all configured I/O Processors. MOSTEST allocates up to 256 512-word buffers. MOSTEST writes to and reads from each buffer using block sizes of 1 to 512 words with varying data patterns.

The station error display lists any errors discovered. To obtain the error display, enter the ERROR command at the station console. The error display gives an address, the data expected, and the data received. MOSTEST displays a PASS COMPLETE message on the Kernel console of each IOP involved when all of allocated Buffer Memory has been tested.

NOTE

MOSTEST cannot be run concurrently with other software.

To load the diagnostic overlay, enter the MOSTEST command at the IOP0 Kernel console as follows:

    MOSTEST

Because the test takes control of the I/O Subsystem (IOS) while it is running, the diagnostic displays the following message on the screen:

    ARE YOU SURE YOU WANT TO RUN THIS TEST?

If the response to the system query is YES, MOSTEST runs.

The diagnostic runs until the STOP MOSTEST command is entered at the console of each I/O Processor in which MOSTEST is running.

8.2.4 XDK

The XDK diagnostic tests the Peripheral Expander 80 Mbyte disk drive. XDK establishes a 4000_8 parcel buffer of data and writes the buffer to the entire disk, eight sectors at a time. The diagnostic then reads the disk into a second buffer and compares the two buffers for errors.

To load the diagnostic overlay, enter the XDK command at the IOP0 Kernel console as follows:

    XDK

The XDK diagnostic displays the following message on the screen:

    THIS TEST WRITES OVER THE ENTIRE DISK - CONTINUE?

Enter Y to continue the test. If any other key is pressed, the test aborts. XDK begins when Y key is entered and runs until completion or an error is encountered.

The IOP0 Kernel console displays error messages when a data compare error is encountered. The error display includes the data expected, the data received, and an option to continue or quit as shown below:

    DATA EXPECTED *xxxxxx*
    DATA RECEIVED *xxxxxx*
    PRESS C TO CONTINUE, Q TO QUIT

XDK displays the following message each time the disk is tested:

    PASS COMPLETE

XDK runs until an error is encountered or until the following command is entered at the IOP0 Kernel console.

    ABORT @DK0

8.2.5 XMT

XMT tests the expander chassis tape drive. The diagnostic writes several multiblock files to the tape drives, reads the files back, and compares the data for errors.

To load the diagnostic overlay, enter the XMT command at the IOP0 Kernel console as follows:

```
XMT
```

The XMT diagnostic displays the following message on the screen:

```
TYPE ANY KEY TO CONTINUE
```

XMT begins when any key is pressed.

The IOP0 Kernel console displays error messages when a data compare error is encountered. The error display includes the data expected, the data received, and an option to continue or quit as shown below:

DATA EXPECTED *xxxxxx*
DATA RECEIVED *xxxxxx*
PRESS C TO CONTINUE, Q TO QUIT

XMT runs until an error is encountered or until the following command is entered at the IOP0 Kernel console:

```
ABORT @MT0
```

8.2.6 XPR

XPR exercises the Peripheral Expander printer. Printer output consists of alternating pages of characters and plots and must be inspected visually to determine if an error occurred.

To load the diagnostic overlay, enter the XPR command at the IOP0 Kernel console as follows:

```
XPR
```

XPR begins executing when the diagnostic is loaded and runs until the following command is entered at the IOP0 kernel console.

```
ABORT PR0
```

PART 2

SOFTWARE MAINTENANCE

INTRODUCTION 1

Individual diagnostics for Cray CPUs and the I/O Subsystem (IOS) are all assembled and supported by using the Cray Operating System (COS) and its assemblers, CAL and APML. Job decks must be keypunched or text-edited and submitted to COS through the customers front-end computer or station.

The boots (CPXM/CPUM, IOPM/IOPMA, and BMT) and the Diagnostic Support System (DSS) are all assembled under DSS using its assembler, APAL.

The following topics are discussed in part 2, Software Maintenance:

- Assembling diagnostics
- Installing CPU diagnostics
- Installing IOP diagnostics

ASSEMBLING DIAGNOSTICS 2

Two utilities are used to install and maintain CPU and IOP diagnostics on Cray Computer Systems: BUILD (BLD) and the Expander Chassis Driver (ECD).

2.1 BUILD (BLD)

BUILD (BLD) is a COS utility that is used to build the diagnostic system. BLD processes a list of files using the the FLIST parameter on the BUILD control statement.

2.1.1 BLD CONTROL STATEMENT

The BLD control statement appears as follows:

BLD,FLIST=*filelist*,B=*binary*,L=*listing*,D=*doc*,

LPP=*lpp*,BLIST=*blist*,LLIST=*llist*.

Parameters are listed in keyword format.

FLIST=*filelist*
Dataset name of FLIST; the default is $IN.

B=*binary* Binaries. The options are listed below:

B=DEFER	Binaries are saved to Cray disk only; the default.
B=FDMP	Binaries and text object code to FDMP tape
B=0	No binaries produced

L=*listing*

Listings. The options are listed below:

L=DEFER	Listings are saved to Cray disk only; the default.
L=PRINT	Print listings.
L=0	No listings produced

D=*doc* Documentation. The options are listed below:

D=DEFER	Documentation is saved to Cray disk only; the default.
D=PRINT	Print documentation.
D=0	No documentation produced

LPP=*lpp* Lines per page for listings; the default is 45.

BLIST=*blist*

Name of the dataset binary where file names are written. *blist* is used as FLIST by ECD; the default is file BLIST.

LLIST=*llist*

Name of the dataset listing where file names are writtend. LLIST is used as FLIST by ECD; the default is file LLIST.

2.1.2 FLIST FILE ENTRY

The following conventions must be followed when writing an FLIST file entry:

- Comments lines are allowed, but an asterisk must be placed in column 1 of the statement.
- All parameters are limited to eight characters.
- Not all fields are used for each file type. A field cannot be null. NA should be used as a place holder for the LOP or PL FLIST options on a text file.

The FLIST file entry comes in two formats: format 1 and format 2.

Format 1:

```
type   pl   deck   tfile   fnt   lop
```

FLIST parameters are explained below.

type Indicates how the file is processed; *type* can be one of the following:

APML	APML source file
CAL	CAL source file
TEXT	Text file
INFO	Documentation
CB	Command buffer

pl Names the program library to be accessed. *pl* can be one of the following:

IOPPL	IOP diagnostics and utilities
XMPPL	XMP diagnostics
CRAYPL	CRAY-1 diagnostics (all models)

deck Names *deck* as the deck on the PL to be accessed

tfile Names the destination file on the FDMP tape; such as :IFP.

fnt Indicates the number (2-5) of the destination File Name Table (FNT)

lop Indicates the list option for the assembler; such as MONITOR.

Format 2:

FLIST format 2 allows you to designate the program modules to be copied from an input dataset. A dollar sign in column 1 is interpreted as a build directive. It is possible to change the listing (L), binary (B), or documentation (D) directives as you are processing files. If you want to turn the listing option off for a specific file, the first field should contain a $, the second an L and the third a 0. The following example turns the listing option off on the next directive:

```
$      L       0
CAL    CRAYPL  CMD     :CMD    4      NONE
/EOF
```

The following example defers the listing option on the next directive:

```
$      L       Defer
CAL    CRAYPL  CMD     :CMD    4      NONE
/EOF
```

An FLIST is always terminated by two /EOFs unless UPDATE directives are desired. In that case, the FLIST is terminated by one /EOF and followed by a list of UPDATE directives. For more information on UPDATE, see the UPDATE Reference Manual, CRI publication SR-0013. An UPDATE directive has been added to an FLIST entry in the following example:

```
$          L          Defer
CAL        CRAYPL     CMD         :CMD        4          NONE
/EOF
$          CRAYPL
*D CMD.120
*D SFM.1023
/EOF
$          IOPPL
*D ADB.120
AAA        AB=4                   .SET COUNT
/EOF
/EOF
```

The dollar sign in the first field indicates that the UPDATE directives for the PL given in the second field are present. Directives can be indicated for as many as 10 PLs. The last directive must be terminated by two /EOFs.

2.1.3 BLD JOB CONTROL DECK

The BLD job control deck shown below does the following:

- Uses dataset S983 as a file list of diagnostics to process
- Updates and assembles all diagnostics, command buffers, documentation, and utilities as specified by the file list
- Saves all binaries on COS disk for future processing by ECD. Dataset S983B contains a description of all the binary files and will be used as the FLIST parameter of ECD.
- Saves all Listings on COS disk for future processing by ECD. Dataset S983L contains a description of all the listing files and will be used as the FLIST parameter of ECD.
- The following is an example of a COS job that assembles the diagnostic system:

```
ACCESS,DN=BLD,ID=DIAGSYS.
BLD,FLIST=S983,BLIST=S983B,LLIST=S983L.
CALL,DN=JCL.
/EOF
```

2.2 EXPANDER CHASSIS DRIVER (ECD)

ECD, the Expander Chassis Driver utility, does physical I/O of files to the Peripheral Expander chassis or other front-ends. This utility runs under COS and is used for printing diagnostic listings and writing DSS-compatible tapes.

Format:

```
ECD,F=function,FLIST=filelist,TC=txtcon.
```

Parameters are in keyword form, the only required parameter is F.

F=*function*

Expander Chassis Driver functions; options are:

FDMP	DSS FDMP of FLIST files
PRINT	Print FLIST files.
ECLIPSE	Dispose FLIST files to RDOS station.
CLEAN	Delete all editions of PDNs in FLIST.
AMDAHL	Dispose listings to the Amdahl.

FLIST=*filelist*

Listing files to be processed; the default is $IN. *filelist* must have the following format:

mfile efile fntn df

mfile	Name of file on mainframe
efile	Name of file on Peripheral Expander chassis
fntn	Destination DSS FNT number
df	The following dataset formats are available:

- B Binary
- T Text
- L Listing

TC=*txtconv*

Text conversion from/to DSS/ASCII

TC=Y Convert standard ASCII text to DSS text if destination is some type of DSS tape.

Convert DSS format text files to standard ASCII if destination is mainframe.

TC=N No text conversion

The example job deck shown below, uses dataset S983B as a file list of diagnostic binaries. The object code is written to the Peripheral Expander chassis tape drive as a DSS-compatible FDMP tape.

```
ACCESS,DN=ECD,ID=DIAGSYS.
ECD,F=FDMP,FLIST=S983B.
CALL,DN=JCL.
/EOF
```

The example job deck, shown below, uses dataset S983L as the file list of diagnostics to be printed on the Peripheral Expander chassis printer.

```
ACCESS,DN=ECD,ID=DIAGSYS.     ECD,F=PRINT,FLIST=S983L.
/EOF
```

INSTALLING CPU TESTS 3

To install or modify CRAY CPU diagnostic tests, compose a job consisting of Cray Operating System (COS) control statements on the customer's front-end computer system and submit it to COS. The job accesses the diagnostic program library (UPDATE), assembles the program (CAL), and disposes the binary to the I/O Processor tape drive. The listing is printed after the Diagnostic Support System (DSS) is brought up on the I/O Subsystem (IOS) and the binary is read onto the File Name Table (FNT 4). Monitors do not have to be appended and resaved because they are assembled internally with each test.

3.1 PROGRAM LIBRARY (X200PL)

The sources for all CPU diagnostics are contained on a single program library (PL) that resides on a permanent dataset named X200PL.† The PL is accessed using the UPDATE utility. For more information on UPDATE, see the UPDATE Reference Manual, CRI publication SR-0013. Each program is referred to as a deck on the PL. Special decks, called common decks, are not stand-alone programs, but rather collections of frequently used code that can be inserted within a program with the following directive:

*CALL *name*

The following common decks can be inserted within a program:

- XMTA
- XMTI
- MTX

† This dataset is renamed with each release, X200PL for the 2.0 release, X201PL for the 3.0 release, and so on.

3.2 ASSEMBLING A PROGRAM

Use the customer's front-end operating system to create and submit the job shown below. The following job can be used to assemble or modify any program on the PL. Refer to the UPDATE Reference Manual, CRI publication SR-0013, for information on insert and delete directives.

```
JOB,JN=LWS,T=19
ACCOUNT,AC=acctno.†
ACCESS,DN=$PL,PDN=CRAYPL,ID=DIAGSYS.
UPDATE,Q.
CAL,I=$CPL,DEBUG,LIST=MONITOR.
REWIND,DN=$CPL:$BLD.
ADSTAPE.
ACCESS,DN=IOPTAPE,ID=LOCK,UQ.
DISPOSE,DN=$DS,DC=MT,MF=AP,WAIT,TEXT=DIAGS:0.
DISPOSE,DN=$OUT,DEFER,WAIT,MF=AP.
/EOF
*/††
*C NAME
```

After the job has executed, a tape request is posted on the IOS Kernel console. Mount a scratch tape to receive the program binary and enter the following command at the Kernel console:

```
GO
```

While running under the Diagnostic Support System (DSS), copy the binary onto File Name Table (FNT) 4 using the following commands:

```
DROP name 4
READ @ name 4
```

A new binary, called *name*, is now installed on FNT 4 and can be executed using CMOSX.

3.3 CREATING A NEW PROGRAM

A program source can be assembled by preceding it with control statements and submitting it to the Cray Operating System (COS) as a job. The

† Your local account statement is inserted here. If your site does not use an account statement, delete this entry.

†† Your program name directive should be inserted here.

program file, created under the customer's front-end operating system, should look like the example shown below:

```
JOB,JN=LWS,T=19.
ACCOUNT,AC=acctno.†
ACCESS,DN=$PL,PDN=CRAYPL,ID=DIAGSYS.
UPDATE,Q.
CAL,I=$CPL,DEBUG,LIST=MONITOR.
REWIND,DN=$CPL:$BLD.
ADSTAPE.
ACCESS,DN=IOPTAPE,ID=LOCK,UQ.
DISPOSE,DN=$DS,DC=MT,MF=AP,WAIT,TEXT=DIAGS:0.
DISPOSE,DN=$OUT,DEFER,WAIT,MF=AP.
/EOF
*DECK      TEST
           IDENT     TEST
           ABS
           BASE      M
           ORG       0
*CALL      MTA
           SPACE     4
START      S1        1
           R /MTA/IO   UPDATE PASS COUNT
           J START
*C TEST
```

After the job has executed, a tape request is posted on the Kernel console. Mount a scratch tape to receive the program binary and enter the following command at the Kernel console:

```
GO
```

While running under DSS, copy the binary onto File Name Table (FNT) 4 by entering the following commands:

```
DROP name 4
READ @ name 4
```

A new binary, called *name* is now installed on FNT 4 and can be executed using CMOSX.

† Your local account statement is inserted here. If your site does not use an account statement, delete this entry.

INSTALLING I/O SUBSYSTEM DIAGNOSTICS 4

To install I/O Subsystem diagnostics, compose a Cray JCL file consisting of Cray Operating System (COS) control statements on a front-end computer system and submit the job to the Cray mainframe through COS. The program accesses the diagnostic program library (UPDATE), assembles the program using APML, and disposes the binary to the I/O Processor tape drive. The listing is printed after the Diagnostic Support System (DSS) is brought up on the I/O Subsystem (IOS); the binary is read onto File Name Table (FNT) 3 of the disk. Monitors do not have to be appended and resaved because they are assembled internally with each test.

4.1 PROGRAM LIBRARY (IOPPL)

The sources for I/O Processor (IOP) diagnostics are contained on a single program library (PL) which resides on a permanent dataset named IOPPL. This library is accessed using the UPDATE utility. See the UPDATE Reference Manual, CRI publication SR-0013, for more information about UPDATE. Each program is referred to as a deck on the PL. Common decks are special decks that can be inserted within another program using the following directive:

*CALL *name*

Common decks are not stand-alone programs, but rather collections of frequently used code. The following common decks can be inserted into another program:

ETA	XMTA type monitor for Eclipse (includes GLODEF)
ETI	XMTI type monitor for Eclipse (includes GLODEF)
GLODEF	Global definitions (error count, pass count, etc.)
GRN	Random number subroutine
PTA	XMTA type monitor for PCU Basic (includes GLODEF)
PTI	XMTI type monitor for PCU Basic (includes GLODEF)

4.2 APML

Diagnostics for the I/O Processor are written for the APML assembler. For more detailed information about APML, see the APML Assembler

Reference Manual, CRI publication SM-0036. An APML program starts with an IDENT statement and finishes with an END statement. Tab columns are set at 1, 10, and 35 for the label, assignment, and comment fields, respectively. Column 20 is used as required for certain pseudo-operations. If a comment is included as part of the APML statement, column 35 must contain a period.

Register usage is assigned with the REGISTER pseudo-operation. The label field, if given, is the register number bias.

Location	Result	Operand	Comment
1	10	20	35
000	REGISTER	(AA,AB,AC)	.scratch registers

Channel commands recognized by APML include IOR(0), PFR(1), PXS(2), LME(3), RTC(4), MOS(5), ERA(16), and EXB(17). If any other channels are to be referenced, they must first be declared using the following CHANNEL pseudo-operation:

```
DKA     CHANNEL       24
```

Jump and return instructions are normally assembled in P+d form. If a diagnostic program is longer than 777 words long, assembly errors occur. A solution to this problem is to reserve a register, set it to 0, and allow the assembler to use it to form dd+k jumps. The BASEREG pseudo-operation accomplishes this.

Location	Result	Operand	Comment
1	10	20	35
	REGISTER	ZE	.constant zero
	BASEREG	ZE	.base register for jumps

If labels need to be attached to data items, they should be attached with an asterisk (*). Use of CON and DATA is not recommended unless defining I/O data. Use one of the data forms shown below:

Location	Result	Operand	Comment
1	10	20	35
LABEL	*		
		'ABCDEF'	.ASCII data
	177777		.octal
	ADR1		.address
	D'16		.decimal
	VWD	8/377,8/'A'	.VWD

If ORG or LOC are used, the address must be *nnnn* /4.

Use EQUALS instead of =.

Use <100> instead of BSS 100.

Use <<100>> instead of BSSZ 100.

4.3 ASSEMBLING A PROGRAM

Use the customer's front-end operating system to create and submit the following job. This job can be used to assemble or modify any program on the PL. Most programs contain monitors that do not appear on the listing, unless the LIST=MONITOR parameter is added to the APML statement.

```
JOB,JN=LWS,T=19.
ACCOUNT,AC=acctno.†
ACCESS,DN=$PL,PDN=IOPPL,ID=DIAGSYS.
UPDATE,Q.
APML,I=$CPL,DEBUG,LIST=MONITOR.
REWIND,DN=$CPL:$BLD.
ADSTAPE.
ACCESS,DN=IOPTAPE,ID=LOCK,UQ.
DISPOSE,DN=$DS,DC=MT,MF=AP,WAIT,TEXT=DIAGS:0.
DISPOSE,DN=$OUT,DEFER,WAIT,MF=AP.
/EOF
*/ program name directive
*C NAME
```

After the job has been processed, a tape request is posted on the IOS Kernel console. Mount a scratch tape to receive the program binary and enter the following command at the Kernel console:

```
GO
```

While running under the Diagnostic Support System (DSS) on the I/O Subsystem (IOS), copy the binary onto File Name Table (FNT) 3 by entering the following commands:

```
DSA ECD
DROP name 3
READ @ name 3
```

† Place your local account statement here. If your site does not use an account statement, delete this entry.

A new binary, called *name* is now on FNT 3 and can be loaded and executed. If a new IOP0 deadstart tape is desired, mount a scratch tape with the write enable ring, and enter the following commands (must be done while running under system DSS0):

```
WRITE TBOOT @
WRITEA A0DIR
```

4.4 CREATING A NEW PROGRAM

A program source can be assembled by preceding it with control statements and submitting it as a job to the Cray Operating System (COS). The program file, created under the customers front-end operating system, should look like the program in figure 4-1.

```
JOB,JN=LWS,T=19.
ACCOUNT,AC=acctno.†
ACCESS,DN=$PL,PDN=IOPPL,ID=DIAGSYS.
UPDATE,Q.
APML,I=$CPL,DEBUG)LIST=MONITOR.
REWIND,DN=$CPL:$BLD.
ADSTAPE.
ACCESS,DN=IOPTAPE,ID=LOCK,UQ.
DISPOSE,DN=$DS,DC=MT,MF=AP,WAIT,TEXT=DIAGS:0.
DISPOSE,DN=$OUT,DEFER,WAIT,MF=AP.
/EOF
*DECK      TEST
           IDENT      TEST
           ABS
           BASE       M
           ORG        O
*CALL      PTA
           END
*C TEST
```

Figure 4-1. APML program file

† Place your local account statement here. If your site does not use an account statement, delete this entry.

After the job has been processed, a tape request is posted on the IOS console. Mount a scratch tape to receive the program binary and enter the following command at the Kernel console:

```
GO
```

While running under the Diagnostic Support System (DSS) on the I/O Subsystem (IOS), copy the binary to File Name Table (FNT) 4.

DSA ECD
DROP *name* 3
READ @ *name* 3

The new binary is now on FNT 3 and can be loaded and executed from disk. If the program is also to be used from tape, a new IOP0 deadstart tape must be created while running under DSS0. Modify A0DIR to include the new program binary by entering the following command:

```
EDIT A0DIR NEWDIR
```

Insert the new program name in the desired position on the tape. Terminate the editor, mount a scratch tape, and enter the following commands:

```
WRITE TBOOT @
WRITE NEWDIR
```

APPENDIX SECTION

CPU TEST DESCRIPTIONS A

This appendix contains a list of the diagnostics that test Cray Computer Systems (CRAY X-MP Computer Systems and CRAY-1 Model A, B, C, S or M Computer Systems, a description of the parameter options that are available for altering individual tests, and error information.

The following naming conventions have been established to identify the monitor that a particular diagnostic uses.

- Programs using the MTA monitor are preceded by an asterisk.
- Programs using the MTI monitor are preceded by a colon.
- Programs using the MTX monitor are preceded by an ampersand.
- Programs that require a memory size parameter to be set are preceded by a pound sign. These programs do not specify a particular monitor type.
- Boots and tests that do not use a monitor have names that are comprised of alpha characters only.
- Programs that are not released or supported by the Diagnostic Systems Department (DSD) are preceded by a semicolon.
- Command buffers are preceded by a slash.

A.1 CPU TEST LISTS

This subsection lists and briefly describes the diagnostic test list for the following Cray Computer Systems:

- CRAY X-MP
- CRAY-1 Models A, B, C, S and M

A.1.1 CRAY X-MP COMPUTER SYSTEMS TEST LIST

The following list of diagnostics run under the control of CMOSX Subsystem 0 and are arranged in alphabetical order. More detailed information on each test can be found in the CRAY X-MP Computer Systems Diagnostic Ready Reference Guide, CRI publication HQ-1005.

Table A-1. CRAY X-MP CPU test list

Test	Description
00	Clear Cray image buffer
11	Set Cray image buffer
:ADD	Address add
#AHT	An indexing test
:AMP2	Two processor monitor
:AMT	Address multiply test
*ARA	Address register add
:ARA	Address register add
:ARB	Address register basic
*ARBA	A register zeros and ones
:AREC	Random error correction test
:ARI	A register input paths
:ARM	Address register multiply
:AVC	Vector register chip test
&AVD	Automatic vector chaining test
:AVE	Vector register chip test
*A130	NSC HYPERchannel interface test
:BATS	B to A and T to S data paths
:BAVE	Vector register chip test
*BIT	Basic CPU instruction test
#BJK	B register jumps
:BLX	Base/limit check test
:BRB	B register basic
*BTDMP	Dump B and T registers to memory
:BTRT	B and T register test
&BTV	B and T register block transfers
*CBG	Memory check bit generator
*CHE	Interface echo test (slave)
*CHN	Channel test
*CHT	Jumpered channel test
*CKB1	Check bit test
*CLR	Clear interrupts and memory in both CPUs
&CMX	Random instruction and operand confidence test
&CT	MCU channel check
*EDB	Exchange data basic
*EXD0	Display Exchange Package IOP0

Table A-1. CRAY X-MP CPU test list (continued)

Test	Description
*EXD1	Display Exchange Package IOP1
#EXJ	Exchange jump test (S/N 9 and above)
&EXM	Status register test
:FIND	Utility that searches for a user-defined parcel value
*FPE	Floating-point error test
*IBP0	Instruction buffer parcel 0
*IBP1	Instruction buffer parcel 1
*IBP2	Instruction buffer parcel 2
*IBP3	Instruction buffer parcel 2
:IBR	Instruction buffer test
:IBRX	Instruction buffer test
*IBR0	Instruction buffer test for parcel 0
*IBR1	Instruction buffer test for parcel 1
*IBR2	Instruction buffer test for parcel 2
*IBR3	Instruction buffer test for parcel 3
&INFC	Interface test
*IFT	Interface test (master)
#JAB	Jump address basic
#JBK	Jump instruction test
*JCB	Jump condition basic
#MCR	Memory column/row test
#MCST	Memory test
#MDSX	Memory test
#MDSZ	Memory test
#MEB	Memory basic
#MIT	Moving inversions memory test
#MIX	Memory test, section selectable from very basic through high-speed stress testing
*MMI	Monitor mode interrupt test
*MNUM	Utility that writes every memory location
:MOVE	Utility that moves data from one area of memory to another
:MPM	Multiprocessor interrupt monitor
:MSEC	Test SECDED modules
#MSL	Memory scope loop
*MTA	Monitor mode monitor
#MTEX	CRAY X-MP version of the MTE monitor
#MTEX16	CRAY X-MP version of the MTE monitor for 16K chips
:MTI	Low level interrupt driven monitor
&MTX	Sophisticated monitoring system
#MVWR	Memory vector write recovery test
#MVX	Vector read/write memory test
*PCI	Programmable clock interrupt
&PDP	PDP 11/70, VAX 11/750, 780 interface
:PORT	A, B, C port test

Table A-1. CRAY X-MP CPU test list (continued)

Test	Description
*RTC	Real-time clock
:RUN	Multiprocess 16 tests
:RUNX	Multiprocess 16 tests per CPU (32 tests total)
:SAR	Random scalar add test
*SCAT	Channel test
:SCN	Scan memory
:SEM	Semaphore test and test and set check
:SEM0	Semaphore test and test and set check for single processor
&SFA	Simulate floating-point add
&SFM	Simulate floating-point multiply
&SFR	Scalar floating-point register
&SIS	Scalar instruction simulator
&SMU	Symmetric multiply
&SR1	1-parcel instruction register conflicts
&SR1-F	1-parcel instruction register conflicts minus floating-point instructions
&SR2	1- and 2-parcel instruction register conflicts
&SR2-F	1- and 2-parcel instruction register conflicts minus floating-point instructions
&SR3	Random register conflicts
*SRA	Scalar integer sum and difference
:SRA	Scalar integer sum and difference
:SRB	Scalar register basic
*SRBA	S register ones and zeros
:SRBA	S register ones and zeros
*SRDP1	Shared register dump utility for cluster 1
*SRDP2	Shared register dump utility for cluster 2
*SRDP3	Shared register dump utility for cluster 3
*SRL	Scalar register logical
:SRL	Scalar logical functional unit
&SRR	Scalar shift instruction
*SRS	Scalar register shift
*SSDIO	SSD I/O test
&STAN	Standard answer test
&SVC	Scalar and vector compare
:TDM	TD mode display memory
:TD00	Test Dead (TD) mode test instruction buffer parcel 0 with zeros
:TD01	TD mode test instruction buffer parcel 1 with zeros
:TD02	TD mode test instruction buffer parcel 2 with zeros
:TD03	TD mode test instruction buffer parcel 3 with zeros
:TD10	TD mode test instruction buffer parcel 0 with ones
:TD11	TD mode test instruction buffer parcel 1 with ones

Table A-1. CRAY X-MP CPU test list (continued)

Test	Description
:TD12	TD mode test instruction buffer parcel 2 with ones
:TD13	TD mode test instruction buffer parcel 3 with ones
*TLT	A and S register loading
#TMX	Timing test
#TMXS	Timing test
:TPM	Test monitor privileged instructions
:TRB	T register basic
:VCH	Vector chaining
:VCTST	V register test
*VDMP	Vector register dump
&VLOG	First and second vector logical test
&VLT	Vector logical test
&VPOP	Vector population count
&VPT	Vector data paths
&VRA	Vector register add
&VRB	Vector register basic
&VRL	Vector register logical
&VRN	Vector random
&VRN-F	Vector random minus floating-point instructions test
&VRR	Vector random with random length and increments
&VRS	Vector register shift
&VRX	Vector register test
*XAE1	Test SECDED checklist generation
*XHG1	Performance monitor test
XMPERR	Error correction test
*XSSD	Channel test
*XZE1	ZE module SECDED test
*XAE2	SECDED check
:ZETST	Check 3ZE module
*ZSDX	SECDED logic and memory test

A.1.2 CRAY-1 COMPUTER SYSTEMS TEST LIST

Table A-2 lists the CPU diagnostics for CRAY-1 Model A, B, C, S and M Computer Systems run under the control of CMOSX Subsystem 0. The tests are arranged in alphabetical order. More detailed information on each test can be found in the CRAY-1 Computer Systems Diagnsotic Ready Reference Guide, publication HQ-1004.

Table A-2. CRAY-1 CPU test list

Test	Description
00	Clear CPU image buffer
11	Set CPU image buffer
*A130	NSC HYPERchannel interface test
:ADD	Address add
:ADRT	DCU buffer address test
#AHT	An indexing test
:AMT	Address multiply test
*ARA	Address register add
:ARA	Address register add
:ARB	Address register basic
*ARBA	A register zeros and ones
:AREC	Random error correction test
:ARI	A register input paths
:ARM	Address register multiply
:BATS	B to A and T to S data paths
&BET	Buffer echo test
*BEU	DD-xx9 channel module test
*BIT	Basic CPU instruction test
#BJK	B register jumps
:BLA	Base/limit check test
:BRB	B register basic
*BTDMP	Dump B and T registers to memory
&BTRT	B and T register test
&BTV	B and T register block transfers
*CAET	CA and CL channel pair register test
*CBG	Memory check bit generator
*CCI	Clear Cray interrupt utility
*CCM	Clear Cray memory utility
*CFLW	CTSS disk flaw utility
*CHANL	
*CHE	Interface echo test (slave)
*CHN	Channel test
:CHR	Interface echo test (RUN monitor)
*CHT	Jumpered channel test
*CHU	Channel test (module types)
*CKB1	Basic check bit storage
*CLR	Clear CPU interrupts and memory
&CLEAR	Clear interrupts and memory in both CPUs
&CMD	Random instruction and operand confidence test
*CT	MCU channel test
*DBRR	Disk beater RUN version (DCU-3)
*DBTR	Disk beater (DCU-3)
*DDR0	A and S register dead dump 0's test
*DDR1	A and S register dead dump 1's test

Table A-2. CRAY-1 CPU test list (continued)

Test	Description
*DFC	Disk fire code (DCU-3)
*DFLW	Disk flaw utility (DCU-3)
*DKRX	Microcode interpreter
*DKSE	Disk aid interpreter
*DSKR	Basic disk test RUN version (DCU-3)
*DSKZ	Basic disk test
*EDB	Exchange data basic
*EJT	Exchange mechanism test (serial number (S/N 8 and below)
*EXD	Examines Exchange Package parameters for time out programs
#EXJ	Exchange jump test (S/N 9 and above)
&FCS	Write function test
*FCU	Fire code test
&FLAC	CTSS flaw address calculator
*FPE	Floating-point error test
&FUTA	Add functional unit reliability test
*IBP0	Instruction buffer parcel 0 test
*IBP1	Instruction buffer parcel 1 test
*IBP2	Instruction buffer parcel 2 test
*IBP3	Instruction buffer parcel 3 test
:IBR	Detect instruction buffer failures test
*IFT	Interface test (master)
#JAB	Jump address basic
#JBK	Jump instruction
*JCB	Jump condition basic
#MCR	Memory column/row test
#MCST	Memory test
#MCST1	Memory test
#MDSX	Memory test
#MDSZ	Memory test
#MEB	Memory basic
#MIT	Moving inversions memory test
#MIX	Memory test, section selectable from very basic through high-speed stress testing
*MMI	Monitor mode interrupt test
&MPC	Memory parity circuit test
:MPM	Multiprocessor monitor
:MSEC	Test SECDED modules
*MSEE	Utility that performs repeated outputs of CPU through the 6 Mbyte channel
#MSL	Memory scope loop
*MSL0	Memory scope loop for processor 0
*MSLR	Memory scope loop for right bank

Table A-2. CRAY-1 CPU test list (continued)

Test	Description
*MTA	Monitor mode monitor
:MTD	Monitor (8-bank Cray Computer System)
#MTE	Monitor
*MTE	Monitor
:MTI	Low level interrupt driven monitor
&MTS	Comprehensive monitor (monitor and interrupt mode driven)
#MVWR	Memory vector write recovery test
#MVX	Vector read/write memory test
*PADI	SH (input control) module address register test
*PADO	SI (output control) module address register test
*PCI	Programmable clock interrupt
*PDP	PDP 11/70, VAX 11/750, 780 interface
*RTC	Real-time clock
:RUN	Multiprocess 16 tests
:RUN8	RUN monitor for 8-bank CPUs
:SAR	Random scalar add test
*SCAT	Channel test
:SCN	Scan memory
#SECD	SECDED memory test
&SFA	Simulate floating-point add
&SFM	Simulate floating-point multiply
&SFMR	Simulate floating-point mulitpy random
&SFR	Scalar floating-point register
&SIS	Scalar instruction simulator
&SMU	Symmetric multiply
&SR1	1-parcel instruction register conflicts
&SR1-F	1-parcel instruction register conflicts minus floating-point instructions
&SR2	1- and 2-parcel instruction register conflicts
&SR2-F	1- and 2-parcel instruction register conflicts minus floating-point instructions
&SR3	Random register conflicts
*SRA	Scalar integer sum and difference
:SRA	Scalar integer sum and difference
:SRB	Scalar register basic
*SRBA	S register ones and zeros
:SRBA	S register ones and zeros
#SRIO	Register conflict test for problems that occur in the operating system environment
*SRL	Scalar logical functional unit
:SRL	Scalar logical functional unit
&SRR	Scalar shift instruction
*SRS	Scalar register shift

Table A-2. CRAY-1 CPU test list (continued)

Test	Description
&SSFR	Simulate floating-point reciprocal
:STAN	Standard CPU answer test
&SVC	Scalar and vector compare
*TDI0	TD mode test instruction buffer parcel 0 with ones
*TDI1	TD mode test instruction buffer parcel 1 with ones
*TDI2	TD mode test instruction buffer parcel 2 with ones
*TDI3	TD mode test instruction buffer parcel 3 with ones
:TDM	TD mode display memory
:TDMP	Memory address test (0-1000)
*TLT	A and S register loading
#TMX	Timing test
#TMXS	CPU timing test
:TPM	Test monitor privileged instructions
:TRB	T register basic
*VCH	Vector chaining
:VCTST	V register test
*VDMP	Vector register dump
&VLOG	Second vector logical test
&VLT	Vector logical test
&VPOP	Vector population count
&VPT	Vector data paths
&VRA	Vector register add
&VRB	Vector register basic
&VRL	Vector register logical
&VRN	Vector random
&VRN-F	Vector random minus floating-point instructions test
&VRR	Vector random with random length and increments
&VRS	Vector register shift

A.2 PARAMETERS

All diagnostics, except online, use monitors and recognize certain parameters that control test execution. These parameters can be set in the data information block (DIB). The @STOP parameter, for example, alters the way a test runs depending on the value that is stored in @STOP. The following options are available for @STOP:

Address	Value	Description
@STOP	NSOE (26=0)†	No stop on error; the test runs continuously and accumulates pass and error counts.
@STOP	SOE (26=1)†	Stop on error; the test stops when an error is encountered.
@STOP	LOE (26=2)†	Loop on error; the test loops on error and accumulates pass and error counts on the first error encountered.
@STOP	SLOE (26=4)†	Scope loop on error; the test scope loops on the error.

Certain locations identify the hardware the diagnsotic is testing. The following addresses identify local hardware configuration for diagnostic testing:

Parameter	Description
@MLAST (70)†	Sets limit address; the following limits can be specified:

Chip size	Address	Size
1000000	777777	1/4 Million word (16 banks)
2000000	1777777	1/2 Million word (16 banks)
4000000	3777777	1 Million word (16 banks)
10000000	7777777	2 Million word (16 banks)
20000000	17777777	4 Million word (32 banks)

(1003)† Bank to be tested

0-37=Test specified bank
40=Test all banks

@BANKS (71)† Number of banks

$20=20_8$ banks
$40=40_8$ banks

@CHIP (1005)† Chip size

12=1K chips
14=4K chips
16=16K chips

† Hard addresses are subject to change at any time. Standardized addressing using the Data Information Block (DIB) replaces hard addresses with address parameters.

Table A-3 lists the addresses of the test parameters used by the DIB.

Table A-3. Data information block (DIB)

Parameter	Address†	Description
@NAME		Diagnostic name
@REV		Diagnostic revision
@LEVEL		Diagnostic level
@TARGET		Target CPUs
@SECS		Section select
@STOP		Stop conditions
@REPEAT		Repeat conditions
@MODE		Special mode
@SECC		Current section
@PASS		Pass count
@ERROR		Error count
@CODE		Error code
@ACT		Actual data
@EXP	22	Expected data
@DIF	20	Difference
@ERA	25	Error address
SLZ		Suppress leading zeros
TXT		Text display
IND		Indirect value
@MLAST	70	Memory last Address
@BANKS	71	Number of banks
@CHIP	1005	Chip size
@CPUT	57	Type of CPU
@CPUN		Number of CPUs
@CPUSN		CPU serial number
@OPEN1		Open location
@OPEN2		Open location
@MTRT	60	Monitor type
@MCUT	61	MCU type
@MCUI	63	MCU input channel
@MCUO	64	MCU output channel
@MCUF	65	MCU first word address
@MCUW	66	MCU word count
@MET		Common
$NAME		Name of diagnostic
$REV		Revision of diagnostic
$LEVEL		Level of diagnostic

† Hard addresses are subject to change at any time. Standardized addressing using the Data Information Block (DIB) replaces hard addresses with address parameters.

Table A-3. Data information block (continued)

Parameter	Address†	Description
$TARGET		Target CPU's
$SECS	43	Section select
$STOP	26	Stop conditions
$REPEAT	27	Repeat conditions
$MODE		Special mode
$SECC	44	Current section
$PASS	24	Pass count
$ERROR	23	Error count
$CODE	31	Error code
$ACT	21	Actual data
$EXP	22	Expected data
$DIF	20	Difference
$ERA		Error address
@FINST	32	Failing instruction
@FI		Failing I (Result)
@FJ		Failing J operand
@FK		Failing K operand

† Hard addresses are subject to change at any time. Standardized addressing using the Data Information Block (DIB) replaces hard addresses with address parameters.

Table A-4 lists the CPU type options that can be entered in location @CPUT.

Table A-4. CPU types

Parameter	Address	Description
CPA	1	CRAY-1/A
CPS	2	CRAY-1/S
CPX	4	CRAY X-MP
CPM	10	CRAY-1/M
CPXM	20	CRAY X/M
CPX4	40	CRAY X-MP4

Table A-5 lists the stop conditions that can be entered in location @STOP.

Table A-5. Stop conditions

Parameter	Value	Description
SOE	1	Stop on error (default)
NSOE	0	No stop on error;
SSC	10	Stop at end of subcondition
SCE	20	Stop at end of condition
SSS	40	Stop at end of subsection
SES	100	Stop at end of section
SET	200	Stop at end of test
TRM	Not available	Terminate online test

Table A-6 lists the repeat conditions that can be entered in location @REPEAT.

Table A-6. Repeat conditions

Parameter	Value	Description
NRT	0	No repeat (default)
CONT	1	Continue to next stop
LOE	2	Loop on error;
SCOP	4	Scope loop
RSC	10	Repeat subcondition
RCD	20	Repeat condition
RSS	40	Repeat subsection
RSE	100	Repeat section
RPT	200	Repeat test
RST	400	Reset online test

Table A-7 lists the DIB montior types† that can be entered in location @MTRT

† Deferred implementation

Table A-7. DIB Monitor types†

Type	Value	Description
M0	1	Stand-alone monitor
M1	2	M1 monitor
M2	4	M2 monitor
M3	10	M3 monitor
M4	20	M4 monitor

A.3 ERROR INFORMATION

Most diagnostics execute continuously, accumulating pass counts, and repeating themselves until an error condition is detected. Upon failure, programs give certain status information; usually stored in locations 20_8-40_8. The amount of information can vary, but the following addresses have been defined for error reporting:

Address	Value	Significance
$DIF	20	Logical difference
$ACT	21	Actual data
$EXP	22	Expected data
$ERROR	23	Error count
$PASS	24	Pass count
$SECC	25	Address or section number

Actual memory locations have been replaced by parameter names to reduce the impact of revisions and bugfixes on program maintenance. Knowing the value of the parameter, rather than its location, is also generally more helpful to you in real-life situations. See the Diagnostic Programmer's Guide, CRI publication CP-1006 for detailed information on error parameters and messages.

† Deferred implementation

I/O SUBSYSTEM TEST DESCRIPTIONS B

This appendix contains a list of the diagnostics (table B-1) that test the I/O Subsystem (IOS), a description of the monitors that control IOS diagnostic tests, and error information.

B.1 TEST LIST

Table B-1 lists the diagnostics that run under the control of CMOSX Subsystem 0 and test the I/O Subsystem (IOS). More detailed information for each test can be found in the I/O Subsystem (IOS) Diagnostic Ready Reference Guide, CRI publication HQ-1007.

Most diagnostics contain a common group of instructions called a monitor. The monitor periodically copies its processor's Local Memory into Buffer Memory where the image can be viewed on the Buffer Memory display (part 1, section 3). A monitor, called PTA, performs this task. Another more complicated monitor, called PTI, enables and reports interrupt events by storing the interrupt channel number at address 47_8. The following naming conventions have been established to identify the monitor that a particular diagnostic uses.

- Programs using the PTA monitor are preceded by an asterisk.
- Programs using the PTI monitor are preceded by a colon.
- Programs using any other monitor are preceded by an @.

Table B-1. I/O Subsystem (IOS) test list

Test	Description
*ACT	AMPEX console test
*ADB	Adder basic test
:ADB	Adder basic test
@ADC2	AMPEX disk controller 1 sector deadstart test
@AMPT	Advanced memory parity test
@AMT	Advanced memory test

Table B-1. I/O Subsystem (IOS) test list (continued)

Test	Description
@BCD	Display utility
*BMDG	IBM channel diagnostic
*BMEX	Block multiplexer exerciser
*BMI	Buffer Memory interference
@BMT	Buffer Memory test
@BMTT	Block multiplexer
@BP0	Basic instruction test for IOP0
@BPX	Basic processor instruction test
*CDR	Card reader test
@CFLP	Flaw table utility
@CHMX	Low-speed (LSP4) channel test
*CMB	Local memory basic test
*CMI	CPU memory interference
@CMPAR	Compare memory for parcel match utility
CONDAT	Configuration data
:CTB	Command test basic
@CTM	Command test random
@DBTP	Disk beater
DEADA	Simple deadstart check (TD mode)
DEADB	Simple functional test
DEADC	Simple functional test
DEADD	Simple functional test
DEADE	Simple functional test
DEADF	Simple functional test
DEADG	Simple functional test
DEADH	Simple functional test
DEADI	Simple functional test
DEADJ	Simple functional test
DEADK	Simple functional test
DEADL	Simple functional test
@DFLP	Disk flaw utility
*DIS	Echo display test
*DMAT	Disk DMA test
@DMUS	Disk multiple unit surface analysis
@DMUX	Disk multiple unit exerciser
*DSK	Disk aid routine
@DSKM	Multiple disk aid routine
*DSKP	Disk diagnostic and utilities
*EAT	Expander align tape
@EODMP	Dump exit stack and operand registers
*ESB	Exit stack basic
@EXBAD	Expander box aid utility
*EXD	AMPEX display expander test
*EXT	Extra functions test

Table B-1. I/O Subsystem (IOS) test list (continued)

Test	Description
*GEN	Error generator for error logger
*HICHU	High speed
*HISP	Test 100 Mbyye channel to CPU
@HSB	High-speed basic
*HSSD	SSD test that runs from the I/O Processor on the 100 Mbyte channel
*HSPGN	100 Mbyte channel parity generator
*HSQ	High-speed quick
*IA130	NSC HYPERchannel interface (hit) test
*IFP	Interface test
*IFPDP	Interface test
@IMON	Interrupt monitor
*INT	Interrupt address test
IPC	Interprocessor channel boot
*ISS	Instruction issue test
*LMA	Local Memory address test
*LMB	Local Memory basic
:LMB	Local Memory basic
@LMP	Local Memory parity
@LMT	Local Memory test
*LPT	Line printer test
:MAGR	Mag tape reliability test
*MCUC	MCU channel test
*MDS	Memory data test
:MDS	Memory data test
@MDS0	Test section 0 of Local Memory
@MDS1	Test section 1 of Local Memory
@MDS2	Test section 2 of Local Memory
@MDS3	Test section 3 of Local Memory
*MDSP	Memory parity test
*MDSX	Memory data and address test
:MDSX	Memory data and address test
*MDSY	Memory data and address bit test
:MDSY	Memory data and address bit test
@MRA	Set address pattern into memory and registers
@MRC	Clear memory and registers
@MRS	Set memory and registers to all zeros
ONES	Used to set PCU1 deadstart buffer to all ones
*ORA	Operand register addressing
*ORB	Operand register basic
:ORB	Operand register basic
*ORM	Operand register memory test
:ORM	Operand register memory test
*PFPG	Test low-speed channel

Table B-1. I/O Subsystem (IOS) test list (continued)

Test	Description
*PNPO	PN/PO channel loop back test
*PRT	P register test
*PTA	Simple monitor (similar to XMTA)
:PTI	Monitor that handles interrupts (similar to XMTI)
*RJB	Return jump basic
*SDGEN	Error generator for error logger uses 100 Mbyte channel
*SHB	Shifter basic
:SHB	Shifter basic
:SRX	Random operand and instruction test
*STA	Stack test
*TTT	AMPEX display test

B.2 ERROR INFORMATION

Most diagnostics execute continuously, accumulating pass counts and repeating themselves until an error condition is detected. Upon failure, programs give certain status information. The amount of information can vary, but the following addresses have been defined for error reporting:

Address	Significance
2	Keyboard channel (BMT)
3	Channel 3 Local Memory error status
4	Channel 16 error log status
5	Error log first parameter word
6	Error log second parameter word
7	Error log third parameter word
10	Processor number (preset by MCU)
11	Upper address of the common memory image area (preset by MCU) 200 - IOP0 image area 400 - IOP1 image area 600 - IOP2 image area 1000 - IOP3 image area
12	Current test section
13	Current test subsection and condition
14	Error code
15	Lower address (2^0-2^8) of the failing word (CPU or BM)
16	Upper address (2^9-2^{24}) of the failing word (CPU or BM)
17	LM address of the data error

Address	Significance
20	Logical difference
21	Equal to actual data
22	Expected data
23	Error counter
24	Pass count
25	Program address where the error occurred
26	If mode is equal to: 000000 Continue after error, accumulate pass and error count 000001 Stop on error 000002 Report and loop on error 000004 Scope loop; no error reporting. 100000 Do not generate new operands
30	Current accumulator
31	Current P register
32	Current B register
33	Current exit stack pointer
34	Starting accumulator value (included in tests using PTI monitor)
35	Starting P register value (included in tests using PTI monitor)
36	Current carry flag
41	Interrupted channel number (tests using PTI monitor)
42	Interrupted channel status
201	Keyboard channel (DIS, ACT, BCD, TTT)

B.3 TEST DOCUMENTATION

Complete documentation for individual I/O Processor (IOP) tests can be found in the following sources:

- The preamble of the diagnostic test
- I/O Subsystem (IOS) Diagnostic Ready Reference Guide, CRI publication HQ-1007
- Diagnostic Programmer's Guide, CRI publication CP-1006.

FRONT-END INTERFACE TEST DESCRIPTIONS

C

This appendix identifies the following front-end interface (FEI) diagnostics that execute on the customers front-end computer:

- ACE370 (Amdahl-to-Cray interface test)
- CIT (Cray CDC front-end interface test)
- VAX interface test

C.1 ACE370 - AMDAHL-TO-CRAY INTERFACE TEST

ACE370, the Amdahl-to-Cray interface test, tests the IBM or Amdahl side of the Cray interface. This front-end interface (FEI) diagnostic is distributed in two versions. One version runs under the control of the IBM/MVS operating system, the other under the IBM/VM operating system.

C.1.1 IBM/MVS VERSION

The IBM/MVS job control language for running ACE370 should be entered into a procedure library such as SYS1.PROCLIB. U can be specified as any of the following as long as the device is attached to MVS and varied online:

```
U=CRAYLNK1 ---> DEV 10
 =CRAYLNK2 ---> DEV 110
 =CRAYLNK3 ---> DEV 120
 =CRAYLNK4 ---> DEV 200  (MVS procedure default)
```

The following procedure runs a diagnostic on a Cray FEI:

```
//DIAGNOSE  EXEC      PGM=ACE370
//STEPLIB   DD        DSN=SYS2.UTILITY.LINKLIB,DISP=SHR
//SYSPRINT  DD        SYSOUT=A
//SYSDUMP   DD        SYSOUT=A
//SNAP DD   SYSOUT=A
//CRAY1     DD        UNIT=&U
```

To run ACE370, after it has been stored in the procedure library, enter the following command at the operator's console on the IBM system:

```
S CRAYTEST
```

When the IBM/MVS version of the program starts executing, the following query is sent to the operator's console.

```
01 ** CRI DIAGNOSTIC - DO YOU WISH THE DEFAULT OPTIONS (Y OR N)?
```

Answer this message and any succeeding screen messages by entering the system message number, a blank space, and your response. If you want to run IBM master-Cray slave mode using the default options, reply to message 01 by entering the following:

```
01 Y
```

Answering yes to message 01, runs ACE370 with the following default options enabled:

Option	Description
MODE=2	Cray slave
SECTION=2	Unchained I/O
STOP ON ERRORS=Y	Stop on error is set
LOOP COUNT=9999	Number of program iterations
SCOPE LOOP=N	Scope loop is not set

If you want to change any of the default options, reply to message 01 by entering:

```
01 N
```

MVS displays a series of messages that permit you to reset the program options. The messages are as follows:

```
02 ** CRI SPECIFY MODE REQUIRED (0, 1, 2, OR X)
```

Reply to the message by entering one of the following:

Reply	Description
02 0	Loopback mode
02 1	Cray master mode
02 2	Cray slave mode
02 X	X (or invalid reply) terminates the diagnostic.

```
03 ** CRI SPECIFY SECTION NUMBER (1, 2, OR 3)
```

Reply to the message by entering one of the following:

Reply	Description
03 1	Section 1 runs and issues no op instructions in loopback mode
03 2	Section 2 runs and I/O is unchained
03 3	Section 3 runs and I/O is unchained (valid in loopback mode only).

04 ** SPECIFY LOOP COUNT (0001-9999)

Reply to the request by entering the following:

04 *n*

n must be in the range from 0001 through 9999. Invalid responses cause the request to be repeated. This message is not issued when Cray master mode is effect, because the Cray Computer System keeps track of the loop count. The IBM system merely reflects the count it receives from the Cray Computer System.

05 ** IS STOP ON ERROR REQUIRED (Y OR N)?

If the response is not Y (yes), N (no) is assumed.

06 ** IS SCOPE LOOP MODE REQUIRED (Y OR N)?

If the response is not Y (yes), N (no) is assumed.

To run IBM master mode, load the CHE diagnostic (Appendix A) if the FEI is attached to the Cray mainframe, or the IFP diagnostic (Appendix B) if the FEI is attached to the I/O Subsystem (IOS). Set the parameters, and always start the Cray side first.

Either side can be stopped first. If the Cray side is stopped first, a read/write timeout is sent to the IBM. Stop the IBM side at the operator's console by entering the following:

```
CANCEL CRAYTEST
```

If a dump is required, enter the following:

```
CANCEL CRAYTEST,DUMP
```

C.1.2 IBM/VM VERSION

If the IBM/VM version of ACE370 is being used, the test runs from a user console and requires a valid userID and password on the IBM system.

Log on and attach the channel address the FEI is linked to on the IBM to your userID. Enter the following:

```
LOAD ACE370
START
```

The test begins running when START is entered at the keyboard. The same system messages (Appendix C.1.1), without the system message numbers, that are displayed on the console for the IBM/MVS system are displayed for the IBM/VM version. Responses to IBM/VM system messages are entered as if they were IBM/MVS system messages, but without the system message numbers.

C.2 CIT - CRAY INTERFACE TEST (CDC FRONT-END)

The Cray interface test (CIT) is a standalone diagnostic that tests the CYBER side of the Cray interface. CIT communicates with the IFT or the CHE diagnostic on the Cray side. CIT is released to the field in two forms; as a card deck and as a deadstart tape. To load CIT from cards, set the CYBER deadstart as follows.

Address	Contents
01	75cc
02	77cc
03	e000
04	77cc
05	0001
06	77cc
07	1500
10	2000
11	4277
12	74cc
13	71cc
14	0000

In this list, cc stands for the card reader channel and e represents the card reader equipment number.

To load CIT using tape, set the CYBER deadstart panel as shown below:

Address	Contents
01	75tt
02	77tt
03	e00u
04	77tt
05	0010
06	77tt

Address	Contents
07	1400
10	74tt
11	71tt
12	0000
13	0000
14	0000

In this list, tt stands for the tape channel, E for the tape controller number and U for the tape unit in the content's column shown above.

If the interface is attached directly to a Cray channel, load IFT (Appendix A) in the Cray to run Cray master mode, and load CHE (Appendix A) in the Cray to run CDC master mode. If the interface is attached to an I/O Processor (IOP), the IOP0 deadstart tape must be loaded and DSS started. Press U on the IOP1 console, and boot PCU1. IFP can be loaded and run for both CDC master mode and Cray master mode with the appropriate parameter settings.

To enable a CIT test option, enter the bit value of the desired option. For example, to run CIT under the stop on error condition, set bit 2 of address 1500 equal to 1. Table C-1 lists the test option for the CIT foreign interface test.

Table C-1. CIT test options

Address	Condition	Bit	Description
1500	STOP	2=1 2=2 2=3	Stop on error Stop at the end of the section Stop at the end of the test
1501	REPEAT DELETE	10=1 10=2	Repeat section Delete running message
1502	CHANNEL ADDRESS	6-11 0-10	Channel Must be zeros
1503	BLOCK I/O	0=1	Block I/O; This parameter is not used if loopback mode is in effect.
1504	TRANSFER WORD COUNT		Must be a multiple of 20_8; the the default for loopback mode is 150; the default for master mode is input word count.

Table C-1. CIT test options

Address	Condition	Bit	Description
1505	Not used		
1506	SELECT	9-11	Controller number; usually zero.
		0-8	Zeros
1507	MODE	0=0	CDC master mode
		0=1	Cray master mode
		1=1	Loopback mode
1510	TEST SECTION SELECT	0=1	Section 1; Zeros
		1=1	Section 2; Ones
		2=1	Section 3; Zeros/ones
		3=1	Section 4; Ones/zeros
		4=1	Section 5; Sliding 1 bit
		5=1	Section 6; Sliding 0 bit
		6=1	Section 7; Operator select

NOTE

CIT cannot be run on a CYBER 170/800 series machine that uses microcode to simulate a 12-bit machine.

C.3 VAX INTERFACE TEST

The VAX interface tests (DERT and DER2T) test the VAX side of the Cray interface.

C.3.1 DERT TEST

DERT tests the VAX by running from the VAX to the interface box and from the interface box back to the VAX (loopback mode).

1. Set the front-end interface switches to normal.

2. Insert the floppy diskette (VAX 11/780) or the TU58 and BOOT58 cartridges (VAX 11/750) into the appropriate drives on the VAX computer system.

3. Stop all activity in the VAX (using local parameters) by entering the following command at the VAX operator's console:

   @SHUTDOWN

   Complete VAX shutdown by pressing the following keys:

   CTRL/P (press the CONTROL and letter P keys simultaneously)
   H
   RETURN

4. Load the file from the floppy by entering the following command:

   LOAD DERT.EXE
   RETURN

5. To begin test execution, enter the following:

   ST 100

   Allow the diagnostic to accumulate a few passes (several minutes), and then enter the following commands to halt execution.

   CTRL/P
   H
   RETURN

6. To examine the pass and error counts stored at address 0,4, enter the following:

   E 0/N:1

   If you receive an error count, examine the error buffer by entering the following command:

   E 8/N:5

   This command allows you to look at the error buffer and help aid in the resolution of the problem. The value stored in the address is the error count of the erroring section. The following addresses have been identified for error keeping:

   8 Time out occurred while waiting for data.
   C Number of bytes received is odd, they should be even.
   10 Number of parcels received is not equal to the number of parcels sent.
   14 Data is different.
   18 Time out occurred while waiting for data.
   1C Number of bytes received is odd.

C.3.2 DER2T TEST

DER2T is an interface test that tests the Cray CPU through the interface box to the VAX or tests the VAX through the interface box to the Cray CPU (attached end-to-end). DER2T comes in the following forms:

- VAX master/Cray slave
- Cray master/VAX slave

VAX master/Cray slave

To run DER2T with the VAX acting as the master and the Cray as the slave, use the following procedure:

1. Set the front-end interface switches to normal, and attach the interface test cables end-to-end.

2. If the interface box is connected to a channel on the Cray mainframe, enter the following load command at the Cray Computer console:

   /*PDP

   Set the following parameters:

   1001=1 Cray slave
   1002=0 A access
   1002=1 B access
   1003=*ichan* Input channel address
   1004=*ochan* Output channel address

   Deadstart the Cray Computer System, and set it up to run an idle loop.

3. If the interface box is connected to an IOP channel, enter the following command:

   /IFPDP

   Set the following parameters:

   212=3 PDP interface
   213=2 Cray slave
   234=0 A access
   234=1 B access
   210=*ichan* Input channel address
   211=*ochan* Output channel address

   Deadstart the IOP and set it to run in an idle loop.

4. Insert the floppy diskette for the VAX 11/780 or the TU58 and BOOT58 cartridges for the VAX 11/750 into the appropriate drives on the VAX computer system.

5. Stop all activity in the VAX CPU (using local parameters) by entering the following command at the VAX operator's console:

   @SHUTDOWN

   Complete VAX shutdown by pressing the following keys:

   CTRL/P (press the CONTROL and letter P keys simultaneously)
   H
   RETURN

6. Load the file by entering the following command:

   LOAD DER2T.EXE

8. Start test execution by entering the following:

   ST 100

   Allow the diagnostic to accumulate a few passes (several minutes) and then enter press the following keys to halt execution.

   CTRL/P
   H
   RETURN

9. To examine the pass and error counts stored at address 0,4, enter the following:

   E 0/N:1

   This command allows you to look at the error buffer and aids in the resolution of the problem. The value stored in the error buffer is the error count of the erring section. The following addresses have been identified for error keeping:

   8 Time out; waiting for data.
   C Number of bytes received is odd, they should be even.
   10 Number of parcels received is not equal to the number of parcels sent.
   14 Data is different.
   18 Time out; waiting for data.
   1C Number of bytes received is odd.

Cray master/VAX slave

To run the DER2T interface test with the Cray acting as the master and the VAX as the slave, use the following procedure:

1. Set the front-end interface switches to normal, and attach the interface test cables end-to-end.

2. Insert the floppy diskette for the VAX 11/780 or the TU58 and BOOT58 cartridges for the VAX 11/750 into the appropriate drives on the VAX computer system.

3. Stop all activity in the VAX by entering the following command at the VAX operator's console:

   @SHUTDOWN

4. Complete VAX shutdown by pressing the following keys:

   CTRL/P (press the CONTROL and letter P keys simultaneously)
   H
   RETURN

5. Load the file by entering the following command:

   LOAD DER2T.EXE

6. If the interface box is connected to the Cray channel, enter the following load command at the Cray Computer console:

   /*PDP

   Set the following parameters:

1001=0	Cray master
1002=0	A access
1002=1	B access
1003=*ichan*	Input channel address
1004=*ochan*	Output channel address

   Deadstart the Cray Computer System, and set it up to run an idle loop.

7. If the interface box is connected to the I/O Processor (IOP), enter the following command:

   /*IFPDP

Set the following parameters:

210=*ichan*	Input channel address
211=*ochan*	Output channel address
212=3	PDP interface
213=1	Cray master
234=0	A access
234=1	B access

Deadstart the IOP and set it to run in an idle loop.

8. Deadstart the VAX by entering:

   ST 200

9. Monitor addresses 23 (error count) and 24 (pass count) on the PDP. If an error occurs, PDP listing and error information aid in problem resolution.

10. To examine the pass and error counts at address 0,4, enter the following:

    CTRL/P
    H
    RETURN
    E 0/N:1
    RETURN

    If address 23 contains an error count, check the error buffer by entering the following command:

    E 8/N:5
    RETURN

    Checking the error buffer aids in problem resolution. The value in the error buffer is the error count of the erring section:

8	Time out occurred while waiting for data.
C	Odd bytes
10	Number of parcels received is not equal to the number of parcels sent.
14	Data error
18	Time out occurred while waiting for data.
1C	Odd bytes

DEADSTARTING D

A deadstart can be performed from either tape or disk and requires the setting of various switches. The MASTER CLEAR and DEADSTART switches must be set on the I/O Subsystem (IOS) deadstart panel during either a tape or disk deadstart. Figures D-1 and D-2 show the IOS deadstart panels for Model B I/O Subsystems up through serial number (S/N) 24 and for the models that follow S/N 24, respectively.

The MASTER CLEAR and DEADSTART switches are located on the Power Distribution Unit (PDU) on the Model A IOS. On the Model B IOS, the two MASTER CLEAR (MC) switches on the deadstart panel are labeled SYSTEM MC and IOP0 MC.

Pressing the IOP0 MC switch clears and halts all I/O operations in IOP0. The switch has no effect on the other I/O Processors, Buffer Memory, or error reporting. If the DEADSTART switch is now activated, it deadstarts only IOP0 through the Peripheral Expander channel. All of the other I/O Processors continue to function as before, unless IOP0 issues master clear signals to them through program control.

Pressing the SYSTEM MC switch, halts all IOS functions and clears all I/O Processors, Buffer Memory bank control, and error reporting logic. If the DEADSTART switch is activated, it deadstarts only IOP0; all of the other I/O Processors remain in the master cleared state until deadstarted by IOP0.

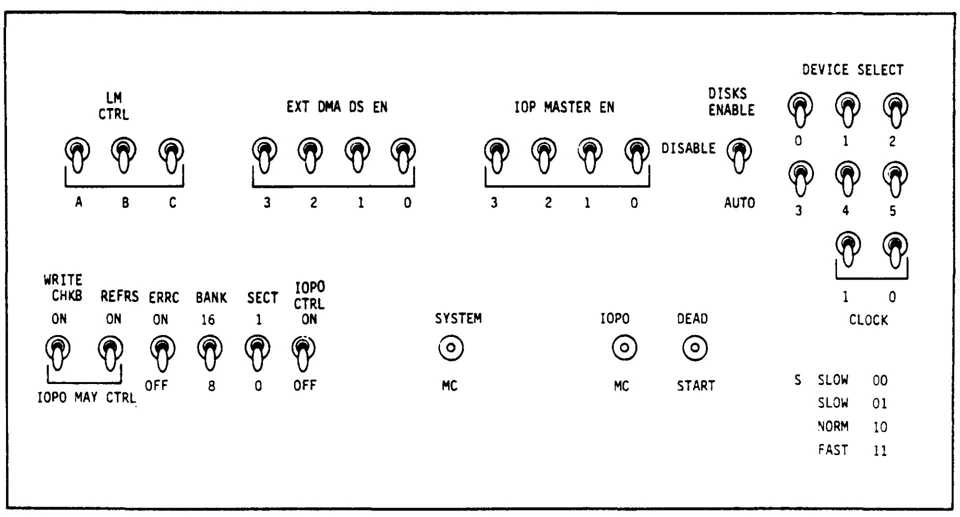


Figure D-1. Deadstart panel controls for IOS Model B through S/N 24

NOTE

The SYSTEM MC switch should not be used when valuable data resides in Buffer Memory, since the switch affects Buffer Memory refresh logic and therefore, is likely to corrupt existing data in the Buffer Memory.

Use the SYSTEM MC switch when the entire system requires initialization; for example, after a power cycle.

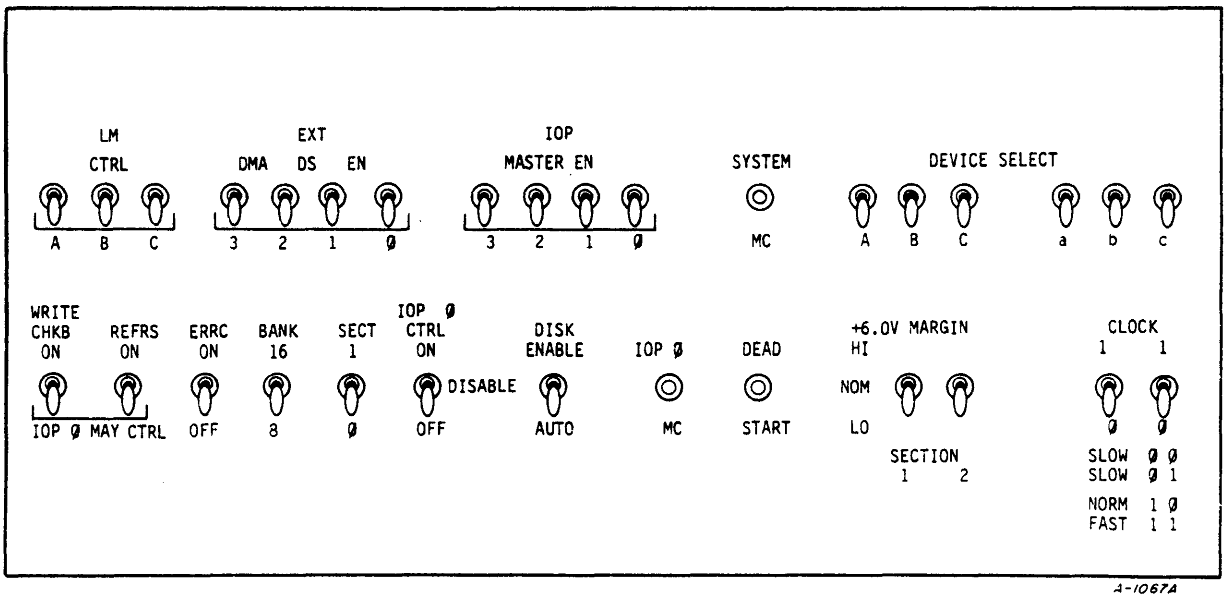
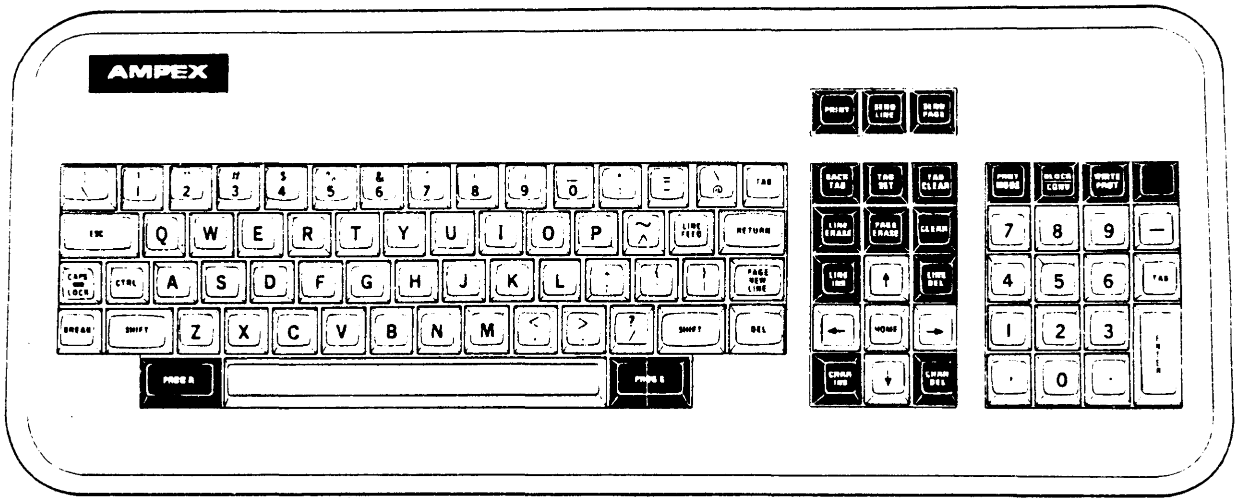


Figure D-2. Deadstart panel controls for IOS Model B above S/N 24

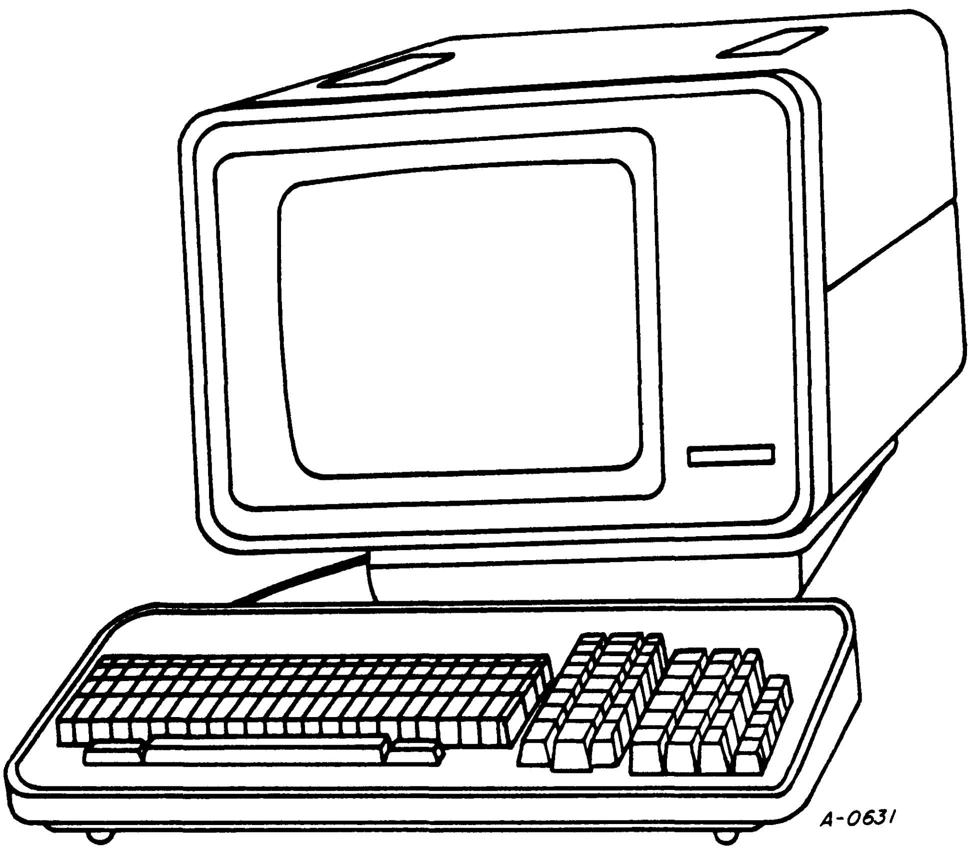
AMPEX (figures D-3 and D-4) or equivalent displays are used for operator interface with the I/O Subsystem and are referred to as CRT-4s throughout this manual. A minimum of two CRT-4s must be included in any configuration (one display is attached to IOP0 and the other to IOP1). Use the following switch settings to configure the switch panel at the rear of the AMPEX console for use with the IOS:

- Set switches 3 and 10 to the ON position (SW1)
- Set switches 1 and 5 to the ON position (SW2)



A-0798

Figure D-3. CRT-4 keyboard



A-0631

Figure D-4. CRT-4 AMPEX display

D.1 TAPE DEADSTART

To perform a deadstart using the deadstart tape supplied by the Diagnostic Systems Department (DSD), follow the steps listed below:

1. Set the following switches on the IOS deadstart panel in the UP position:

   DEVICE SELECT 1 (through S/N 24) or DEVICE SELECT B (above S/N 24)
   DEVICE SELECT 4 (through S/N 24) or DEVICE SELECT b (above S/N 24)
   IOP MASTER EN 0
   IOP MASTER EN 1

2. Mount and ready the IOP0 deadstart tape.

3. On the model B IOS, set the DISK INTERLOCK switch on the deadstart panel to the AUTO position.

4. Press and release the SYSTEM MC/IOP0 MC button on the deadstart panel.

5. Press and release the DEADSTART button on the deadstart panel.

6. Type U on any IOP0 display console (CRT-4) and the boot menu appears (figure D-5).

```
TAPE BOOT MENU

BMT
CBM
CMOSX
CPXM0
DSS0
DUMP
IOPM
IOPMA
IPC
MBUF
SLAT
DKX
```

Figure D-5. Boot menu display

7. Enter the desired selection and press the RETURN key.

NOTE

Terminate each command entry with a RETURN to begin execution. Use the DEL key to backspace over typing mistakes, or use the LINEFEED key to erase the entire line.

D.2 DISK DEADSTART

To perform a deadstart from the 80 Mbyte disk using the deadstart pack supplied by the DSD, follow the steps listed below:

1. Set the following switches on the IOS deadstart panel to the UP position as follows:

   DEVICE SELECT 0 (through S/N 24) or DEVICE SELECT A (above S/N 24)
   DEVICE SELECT 1 (through S/N 24) or DEVICE SELECT B (above S/N 24)
   IOP MASTER EN 0
   IOP MASTER EN 1

2. Mount and ready the current DSS0 deadstart pack.

3. Press and release the SYSTEM MC/IOP0 MC button on the deadstart panel.

4. Press and release the DEADSTART button on the deadstart panel.

5. Type U on any IOP0 display console (CRT-4) and the following message is displayed on the console:

   DSS0 DEADSTART - *release date*

COMMAND BUFFERS E

A command buffer is a stored text file of keyboard commands. When brought into execution, these commands execute as if they were typed in at the keyboard. Two sets of command buffer commands are available: PROCESSOR and INTERACTIVE.

E.1 PROCESSOR COMMANDS

Command buffer PROCESSOR commands are internal and are used for program control and for soliciting input. Table E-1 lists PROCESSOR commands and briefly describes the function of each.

Table E-1. PROCESSOR commands

Command	Description
CALL *lno*	Resumes execution at line number *lno* and saves the return address
CR	Calls return and resumes execution at the return address saved in the CALL instruction
EXIT	Exits the command buffer
GOTO *lno*	Goes to line number *lno* and continues execution at that point
LINK *filename*	Links to command buffer file *filename*
LINK *filename lno*	Ends the command buffer currently in execution and resumes execution with the command buffer *file* at line number *lno*. The default for *lno* is one.
NULL ON	Turns on the null display
NULL OFF	Turns off the null display

Table E-1. PROCESSOR commands (continued)

Command	Description
PAUSE	Pauses indefinitely
PAUSE *cnt*	Pauses for a count of *cnt* milliseconds. If *cnt* is not specified or is a non-numeric character, execution pauses indefinitely or until a LINEFEED is entered. A count of 7777_8 pauses for more than one screen refresh and can be used before a :SNAP command.
RERUN	Reruns the current command buffer
SKIP	Unconditionally skips to the next line
WPC *data*	Waits for pass count (contents of address 24) to equal or exceed data. *Data* can be in the range 1 through 37777777777_8. If error count (contents of address 23) becomes nonzero before pass count equals data, the next line is processed. When pass count equals or exceeds data, the next line is skipped.

E.2 INTERACTIVE COMMANDS

Two commands, INPUT and IF create an interactive command buffer. The commands enable the command buffer to solicit information (INPUT) and select (IF) one of the options based on the input received. Setting up an interactive buffer requires the reservation of eight memory addresses (M0 through M7) for command buffer variables. Each variable may be one character.

Any of the PROCESSOR commands can be entered into an interactive command buffer. Table E-2 describes the INPUT and IF INTERACTIVE commands.

Table E-2. INTERACTIVE commands

Command	Description
INPUT M*v* *n*	Solicits input from the keyboard. The menu variable number *v* is a single alpha code entered by the operator. *n* indicates the octal number of input text lines to be typed.
IF M*v* *o* *a*	Executes the next line if the condition indicated is satisfied or skips the next line if the the condition is not satisfied. *v* is the number of the menu variable (from 0-7), *o* is an operator (= or #), and *a* is any alpha character.

The INTERACTIVE commands in the following example demonstrate how the INPUT and IF statements are used.

```
NULL ON
INPUT M0 7

WHAT IS YOUR CRAY MEMORY SIZE?

  A=0.5 Mbytes
  B=1.0 Mbytes
  C=2.0 Mbytes
  D=4.0 Mbytes

NULL OFF
DX 240

IF M0=A
L 2000000

IF M0=B
L 4000000

IF M0=C
L 10000000

IF M0=D
L 20000000

EXIT
```

In the example above, the INPUT M0 7 sets the input keyboard to menu variable 0 (Cray mainframe Central Memory size). The screen query (WHAT IS YOUR CRAY MEMORY SIZE?) is displayed. If C was entered as a response to the query, Central memory size would be set at 10000000.

REMOTE MAINTENANCE F

I/O Subsystem (IOS) based diagnostics support maintenance from remote terminals over voice grade phone lines. The required hardware consists of a Custom Systems Micro Mutt controller inserted into the console channel line and a 1200 baud modem. Hardware at the remote end consists of a modem and CRT-4 terminal.

While running under DSSO, enter the following command to bring up the Micro Mutt controller:

```
MUTT
```

The MUTT command toggles the display to the 1200 baud rate necessary for Micro Mutt operation. The display maintains the 1200 baud rate, through all subsequent boots, until another MUTT command is entered.

The Custom Systems Micro MUTT controller and Terminal Switch are designated as CTT-1 and CTS-1, respectively. CTT-1 and CTS-1 when configured together allow up to four displays to be remoted to a single remote display for maintenance support. All Cray Computer Systems that are configured with an IOS include CTT-1 and CTS-1 as part of their system configurations.

GLOSSARY

GLOSSARY

A

APAL - A Programming Assembly Language (APAL) is a machine language assembler that is used for MCU modules and boots. APAL runs under the DSS operating system.

APML - A Programming Machine Language (APML) is a machine language assembler that is used for I/O Subsystem (IOS) diagnostics and runs under either the Cray Operating System (COS) or the customer's operating system.

B

Binary - A program module that is in executable form

BIOP - Buffer I/O Processor (IOP1)

Bit position - The location of a bit within a word; counting begins on the left. The sign bit is in bit position zero; the low-order bit is bit 63.

Bit value - The value of bit, assuming the word contains an unsigned integer. The high-order bit, or sign bit, is 2**63; the low-order bit is 2**0. The numbering, counting from the right, can be shown in print as a superscript.

Boots - A collection of stand-alone programs that must be bootstrapped into execution, thus terminating the original program module.

C

CAL - Cray Assembly Language (CAL) is a machine language assembler used for CPU diagnostics. CAL runs under either the Cray Operating System (COS) or the customer's operating system.

Clear (deadstart line) - Set the deadstart line with a 0.

CMOS - Cray Maintenance Operating System for CRAY-1 Model A, B, C, S and M Computer Systems

CMOS/T - Tape version of CMOS

CMOSX - Cray Maintenance Operating System for CRAY X-MP Computer Systems

CMOSX/T - Tape version of CMOSX

Command buffer - A text file that contains a series of keyboard commands.

Correctable errors - Single-bit memory errors

COS - Cray Operating System

CPU - Central processing unit; the CRAY X-MP mainframe.

CPUM - Central Memory test for CRAY-1 Models A, B, C, S and M Computer Systems

CPXM - Central Memory test for CRAY X-MP Computer Systems

Cray master mode - The Cray Computer System builds the buffers and controls the interface test that checks the front-end computer system for accuracy.

Cray slave mode - The front-end computer system builds the buffers and controls the interface test that checks the Cray Computer System for accuracy.

CRT-4 - An Ampex Dialogue 80 display terminal or its equivalent

D

DCU - Disk control unit

DIB - Diagnostic Information Block

DIOP - Disk I/O Processor (an IOP2 or an IOP3 with a DCU)

DIP - Diagnostic Information Packet

DOM - Diagnostic online monitor

Drop (bit) - A track on a disk reports a 0 in a bit location that should contain a 1.

DSD - Diagnostic systems department

DSK - Interpreter for disk aid microcode instructions

DSKM- Interpreter for multiple disk aid microcode instructions

DSS - Diagnostic Support System

DSS0 - The Diagnostic Support System is called DSS0 when it is running in IOP0. DSS0 runs tests on IOP1, IOP2, and IOP3.

DSS1 - The Diagnostic Support System is called DSS1 when it is running in IOP1. DSS1 runs tests on IOP0.

F

FEI - Front-end interface

File - A collection of data stored on disk or tape.

FNT - File name tables are tracks that are used by the Diagnostic Support System to store program files.

H

Hardcore - The minimal hardware assumed to be running in order for a diagnostic to run.

I

IOP - I/O Processor

IOPM/IOPMA - Memory test for the I/O Subsystem (IOS)

J

Job - A sequence of control statements that is used to perform some function with the operating system.

M

MCU - A Maintenance control unit can be either a front-end computer or a tester.

Micro MCU - Micro computer system (Northstar) that is used to run tests on IOP0.

MIOP - Master I/O processor (IOP0)

Monitor - A common block of instructions, assembled into most diagnostics, that usually outputs memory contents to the MCU. It can also trap interrupts or build memory error tables.

MOS memory - Buffer or Central Memory

MTX - Universal CPU test monitor

O

Offline tests - Stand-alone diagnostic tests that are designed to run during dedicated preventive maintenance (PM) periods.

Online tests - Diagnostic tests that run under the Cray Operating System (COS) as regular user jobs.

P

PDU - Power Distribution Unit

Pick (bit) - A track on a disk reports a 1 in a bit location that should contain a 0.

PL - A program library (PL) contains sources for diagnostics.

PTA - A simple monitor (see monitor)

PTI - An advanced monitor (see monitor)

Pulls (deadstart line) - Set the deadstart line with a 1.

R

RTC - Real-time clock

RUN - An advanced monitor which multiprocesses 16 CPU diagnostics

S

Section - A specific test within a larger more general diagnostic test

Source - The assembly language statements for a given program

T

Text - ASCII data such as program source, command buffer, or printer listing

U

Uncorrectable errors - Double-bit memory errors

X

XIOP - Auxiliary I/O Processor (an IOP2 or an IOP3 with a block multiplexer)

INDEX

INDEX

READERS COMMENT FORM

I/O Subsystem (IOS) Based Field Engineer's Diagnostic Reference Manual HM-1002

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME ______________________________

JOB TITLE ______________________________

FIRM ______________________________

ADDRESS ______________________________

CITY ______________ STATE ________ ZIP ________

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention:
PUBLICATIONS

1440 Northland Drive
Mendota Heights, MN 55120
U.S.A.

FOLD

STAPLE

READERS COMMENT FORM

I/O Subsystem (IOS) Based Field Engineer's Diagnostic Reference Manual — HM-1002

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME ____________________

JOB TITLE ____________________

FIRM ____________________

ADDRESS ____________________

CITY ____________ STATE ________ ZIP ________

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention:
PUBLICATIONS

1440 Northland Drive
Mendota Heights, MN 55120
U.S.A.

FOLD

STAPLE

Cray Research, Inc.
Publications Department
1440 Northland Drive
Mendota Heights, MN 55120
612-452-6650
TLX 298444