Support Tools Manual (Licensed Users)
Volume 1, SPU

PA-RISC Computer Systems

HEWLETT
PACKARD

HP Part No. 5960-3149
Printed in USA December 1992

Edition 1 - E1292
FOR HP LICENSED USE ONLY

Notice

System Technology Division
19483 Pruneridge Avenue
Cupertino, CA 95014

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

First Edition . . . . December 1992

List of Effective Pages

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. No information is incorporated into a reprinting unless it appears as a prior update.

All . . . . December 1992

Safety and Regulatory Information

Safety Considerations

This documentation must be reviewed for familiarization with safety markings and instructions before operation. The following figures show some of the safety symbols used in the documentation to indicate various safety considerations.

Warning

The WARNING sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not done correctly or adhered to, could result in injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.

Caution

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not done correctly or adhered to, could damage or destroy part or all of the product. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

Preface

This manual contains information about the online SPU diagnostics currently available for the PA-RISC computer systems. It is intended to be used as technical support hardware documentation. The procedures and software described are focused primarily on the hardware troubleshooting environment and require specific training for correct and safe usage.

Contents

1

Online Diagnostics Overview

Introduction

The online diagnostics subsystem provides a means of testing hardware modules and devices attached to the HP Precision Architecture RISC computer system using either the MPE/iX or HP-UX operating system. The PA-RISC system and the diagnostic system are intimately tied together for error logging, auto-diagnostics, and restricting access to other users during diagnostic testing.

The subsystem provides a common, standard user interface to all the diagnostic programs and utilities, as well as a controlling mechanism for diagnostic access to I/O devices on the system. It can also execute auto-diagnostics (MPE/iX only) for I/O modules or devices generating catastrophic errors. Finally, the subsystem can control the normal I/O error logging process, allowing dynamic display of errors as they occur.

Each major hardware component or aspect of PA-RISC system can be tested by a diagnostic. Each diagnostic is described separately in the following chapters. All of the Diagnostic/Utility chapters share the same format.

Introduction	Brief explanation of the purpose and nature of the diagnostic.
Defects and Enhancements	STARS Database product number for comments about the diagnostic.
Minimum Configuration	Necessary hardware and software to run the diagnostic.
Auto-Diagnostics	The sections that are automatically executed when the MPE/iX system encounters a catastrophic error.
Operating Instructions	Information about how to start the diagnostic.
Default Tests	Lists the tests that are automatically executed if no test sections are specified in the **RUN** command.
RUN Command	Sample **RUN** command using the typical loaded system configuration described in this chapter.
Test Execution	What happens after the diagnostic is started.
Test Section Descriptions	What is being tested by each section of the diagnostic.
Commands	Commands available during the diagnostic (if any).
Error and Warning Messages	Lists all error and warning messages displayed by the diagnostic along with a probable cause and suggested action statements.

Operating Requirements

In order to support the Online Diagnostic subsystem, a PA-RISC computer system must be up and running. User access to at least one functioning terminal is also required.

Online Diagnostic Subsystem Components

The Online Diagnostic subsystem is composed of the the Diagnostic User Interface (DUI) and the diagnostic programs which can be run using either the MPE/iX or HP-UX operating system.

Diagnostic User Interface

The DUI provides the communication link between the user and the diagnostic system. The DUI provides such functions as sending messages to the user from diagnostic programs, and returning replies to the appropriate section(s) of the online diagnostic software.

Diagnostic Programs

The diagnostic programs are a comprehensive set of software to test the devices and components supported on the HP Precision Architecture RISC computer family.

Diagnostic programs are divided into three groups: diagnostics and verifiers; utilities and tools; and system exercisers. Diagnostics are programs that can determine which field replaceable units (FRUs) are malfunctioning.

Verifiers cannot isolate defective FRUs, but can verify which functions of the device are operating properly. Verifiers can determine probable cause of device failures or aid the user in making such determinations. Some diagnostics and verifiers provide thorough tests of the internal I/O modules as well as complete functional tests and system type tests for peripheral devices.

Utilities and tools provide a means for obtaining system information or performing specific I/O operations. System exercisers provide a means of using (loading) a particular part of the system. These programs provide a way of using system resources under stress conditions that equal or exceed those expected under maximum load.

External excercisers are interactive programs provided for some diagnostics to provide the user with access to the set of internal diagnostics and utilities within a particular device (e.g., an SS/80 disk).

Some diagnostics can only be invoked on systems using either the MPE/iX or HP-UX (see the individual diagnostic chapters for more information).

DUI Modes

The diagnostic system provides three modes of operation for each diagnostic program: disruptive mode, destructive mode, and normal mode. The diagnostic system determines the mode that each diagnostic program is allowed to run in by considering such things as the device being tested, whether the program is to be run in auto-diagnostic mode, and the user mode that the system is running in. When the diagnostic program requests access to a device, either at program initiation or at some other time, it is told which mode it is to run in via a device control procedure.

In general, the diagnostic is usually granted destructive mode unless the selected device is a system disk or exclusive access to the device cannot be obtained for the diagnostic. The diagnostic program must decide which tests can be run in the mode it was given.

Disruptive Mode	In disruptive mode, the program can run tests of a "disruptive" nature on the selected device. A disruptive test does not destroy any data on the device, but could cause errors for other users on the system. For example, the internal selftest on a system disk is disruptive, since the disk temporarily goes offline to perform the test, causing errors for others who try to access the disk at the same time.
Destructive Mode	In destructive mode, the program may run any test on the selected device. This mode is required for tests that have the potential for corrupting data on the device being tested. There are virtually no restrictions on tests run in this mode and, therefore, this mode is handled with extreme care by the diagnostic program. An example of a destructive test is one that reformats the media on a system disk, thus destroying all of the data on it.
Normal Mode	In normal mode, the diagnostic program cannot run any tests on the selected device that are potentially destructive or disruptive in nature.

User Modes

There are three user modes available: Single User Mode, Multi-User Mode (normal state), and Single Disk Mode.

Single User	Can be selected by a user with the required capability. The primary purpose of Single User Mode is for testing that may cause data integrity problems. Typically, it is used only in the event of a major problem with the system hardware.
Multi-User Mode	Can be selected by a user with the required capability.
Single Disk Mode	Is selectable only on system boot-up and is system specific.

Security (Standard Systems)

Four levels of security are available for users in the Online Diagnostic subsystem. Access to the various programs is restricted by security level. In addition, each program may restrict certain functionality to users of various security levels.

Level 0 The highest security; the user may install, remove, or update programs through a utility program and may do anything that a user at level 1 may do.

Level 1 The user may perform destructive tests, read or modify data on any device, may enter SUM or MUM modes, and do anything that a user at level 2 may do.

Level 2 The user may perform disruptive tests, but may not display or modify user data, and may do anything that a user at level 3 may do.

Level 3 The user may run non-disruptive tests only.

The following table lists the user capabilities required for each security level for both MPE/iX and HP-UX implementations.

Security	MPE/iX	HP-UX
Level 0	SM,DI	Superuser; Configurable
Level 1	SM,DI,OP	Configurable
Level 2	SM,DI,OP,AM	Configurable
Level 3	all others	Users not in /usr/diag/security

Security (B1 Systems)

The following is a list of the differences between security on standard systems and B1 systems:

- On the standard systems, only the superuser can run diagnostics; on B1 systems, the concept of a superuser is meaningless. Instead, to run diagnostics on a B1 system, it is necessary that the user have **debug** kernel authorization, as well as **mem** subsystem authorization.
- **debug** kernel authorization and **mem** subsystem authorization must be obtained from the Authentication Administrator, who grants authorization via the **authif** program. Since the possessor of **debug** kernel authorization has access to memory and devices, such authorization should be restricted to select users; it should not be added to the user base privileges.
- On B1 systems, the concept of "security levels" is of little value, since the only users who can run diagnostics are those with **debug** and **mem** authorizations, which require the equivalent of security level zero, anyway.
- On B1 systems, all devices must be added by the system administrator, since online diagnostic programs on such systems do not **mknod** devices.

2

Diagnostic User Interface

Introduction

The Diagnostic User Interface (DUI) enables users to run on-line diagnostic programs on PA-RISC systems through a common, consistent interface. Users need only learn one set of commands to run, process, and manipulate all on-line diagnostic programs.

Numerous commands are available to start, stop, monitor, add, and delete diagnostic programs.

Various utility functions are also provided through the DUI - the output of a diagnostic may be redirected to a file and/or echoed to a hardcopy device; the inputs to a diagnostic program may be obtained from a file rather than from the user's terminal; part of a diagnostic session may be controlled from a command file. The complete set of all such functions will be found in the "COMMANDS" section of this document.

All of the commands accessible through the DUI are invoked identically on all systems on which the online diagnostics subsystem runs. Each individual command has the identical effect on every system on which the online diagnostics subsystem runs.

Diagnostic programs which run under the DUI include:

- programs which test peripheral devices and individual circuit boards and attempt to isolate a failure to a FRU (Field Replaceable Unit)
- programs which load (or stress) a system in order to recreate the conditions under which a system might exhibit failures
- programs which can isolate a "functional" failure, such as a read or a write failure, even if they cannot pinpoint the hardware involved in the failure
- programs which perform utility functions useful to a diagnostician such as creating a map of the system configuration or displaying the contents of log files.

The diagnostic programs available through the DUI may vary from system to system, but all diagnostic programs and associated commands are invoked identically no matter what system they run on.

Note

This document discusses the commands

- DEFAULT
- FOREGROUND
- REPLY
- SET
- SHOWDEFAULT
- SHOWPARMS

and refers to a programmatic interface and the ability to run multiple diagnostics using just one **run** command. None of these features is implemented in the first release. If one of the commands listed above is invoked, nothing will be done. The `DUI` prompt will reappear immediately after the command is given.

If more than one diagnostic is named within the **run** command, an error will be printed, and the `DUI` prompt will reappear. The user may then re-enter the command.

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10023.

Passwords and Support Tool Access (HP 9000 Series 800 Computer Systems Only)

Although the operating system encompasses a complete set of support tools, not all are available for general use. Access to some of these tools is restricted to users who have the appropriate license. Still others are restricted for the use of HP support personnel.

Types of Support Tools

There are three types of support tools:

Free Support Tools
: Programs currently used by customers, which are not password-protected. Typically, these are system verifiers and configuration verification utilities (i.e., mapping utilities).

Licensed Support Tools
: Programs useful in isolating problems down to the level of a defective FRU(s). These programs are password-protected, and are licensed to system-specific users and class support providers.

HP Only Support Tools
: Programs which require significant expertise on the part of the user, and which are potentially destructive to data if used improperly. These programs are also password-protected.

Types of Passwords

There are four types of passwords available to support tools users:

RCO/SSO Password
: This password is used by CEs and RCEs, and permits access to the full range of support tools (i.e., free, licensed, and HP only). It is a single password applicable on all systems, which changes every six months. It is valid for a single session, and must be re-enterd for each subsequent session.

Class Password
: This password is used by class support providers, and by system-specific users with large numbers of systems. It allows access to all free and licensed diagnostics. A single password permits access to support tools on all systems of the same class; for example, all HP 9000 Series 800 low-end systems, like the 808, 815, etc. This password also changes every six months. To execute offline diagnostics, the password will have to be entered for each session; for online diagnostics, the password will only have to be entered once every six months.

System-Specific Password
: This password is used by system-specific users. It allows a level of access similar to that afforded by the class password. This password also changes every six months. To execute offline diagnostics, the password will have to be entered for each session; for online

diagnostics, the password will only have to be entered once every six months.

Temporary Password

This password is used by CEs and RCEs, and is provided by them to a customer, in order to obtain customer assistance in troubleshooting. This password will be valid for anywhere from one to twenty days (as defined by the CE or RCE).

Support Tool Banners and Information Messages

The following subsections discuss the different support tool banners displayed by the user's system, and a typical information message displayed when a support tool is invoked.

Support Tool Banners

There are four different banners that the system can display when you access or attempt to access the support tools suite:

1. Initial diagnostic utility system banner (loads when DUI is first run)
2. General customer use banner (loads with each diagnostic)
3. Licensed customer use banner (loads with each diagnostic)
4. HP support personnel use only banner (loads with each diagnostic)

The following banner is displayed whenever you run the DUI:

```
***************************************************************************
******                                                               ******
******                  ONLINE DIAGNOSTIC SYSTEM                     ******
******                                                               ******
******      (C) Copyright Hewlett-Packard Co 1987,1988,1990          ******
******                   All Rights Reserved                         ******
******                                                               ******
******                                                               ******
******                   DUI Version XX.XX.XX                        ******
******                                                               ******
******              Diagnostic Monitor Version XX.XX.XX              ******
******                                                               ******
***************************************************************************
```

The following banner is displayed, whenever you run a support tool which is available for general customer use:

```
*****************************************************************************
******                                                                 ******
******                     Name of Diagnostic                          ******
******                                                                 ******
******        (C) Copyright Hewlett-Packard Co 1987,1988,1990          ******
******                     All Rights Reserved                         ******
******                                                                 ******
******   HP shall not be liable for any damages resulting from the     ******
******   use of this program.                                          ******
******                                                                 ******
******                     Version XX.XX.XX                            ******
******                                                                 ******
*****************************************************************************
```

The following banner is displayed, whenever you run, or attempt to run, a support tool that is restricted for the use of licensed customers or HP support personnel only:

```
*****************************************************************************
******                                                                 ******
******                  Name of Diagnostic                             ******
******                                                                 ******
******        (C) Copyright Hewlett-Packard Co 1987,1988,1990          ******
******                     All Rights Reserved                         ******
******                                                                 ******
******   This program may only be used by HP support personnel and     ******
******   those customers with the appropriate Class license or         ******
******   Node license for systems specified by the license.  HP        ******
******   shall not be liable for any damages resulting from misuse     ******
******   or unauthorized use of this program.  This program            ******
******   remains the property of HP.                                   ******
******                                                                 ******
******                     Version XX.XX.XX                            ******
******                                                                 ******
*****************************************************************************
```

The following banner is displayed, whenever you run, or attempt to run, a support tool whose usage is restricted to HP support personnel only:

```
****************************************************************************
******                                                               ******
******                    Name of Diagnostic                          ******
******                                                               ******
******        (C) Copyright Hewlett-Packard Co 1987,1988,1990         ******
******                   All Rights Reserved                          ******
******                                                               ******
******         THIS PROGRAM IS NOT LICENSED TO CUSTOMERS              ******
******                                                               ******
******   This program is intended for use by trained HP support       ******
******   personnel only.  HP shall not be liable for any damages      ******
******   resulting from unauthorized use of this program.  This       ******
******   program is the property of HP.                               ******
******                                                               ******
******                    Version XX.XX.XX                            ******
******                                                               ******
****************************************************************************
```

Information Messages

If you are a non-licensed user, and you attempt to run a password-protected support tool, a message similar to the following will be displayed:

```
*** WARNING: YOU NEED TO HAVE A self-maintainer/channel LICENSE TO
    RUN THIS PROGRAM.
```

How To Obtain A Password

Changes in Existing Passwords

Passwords and password programs change every six months. Copies of the new password program, and of the RCO/SSO password are sent to Response Center contacts, Country Support Marketing Centers, and CE District Managers for distribution. New passwords will be required of all passworded users on February 1 and August 1 of each year (there is no grace period). The new password will be available from the existing password programs one month before the new password is required.

Programs Which Generate Passwords

There are two programs used to generate passwords:

- VABGEN
- TEMPGEN

The following table presents more information about these two programs:

Program	Run By	Generates	Inputs	User
VABGEN	RC call coordinator or RCEs (non-U.S.)	Customer specific password	Product# (HxxxxA+03D)	Customers with < 20 systems
VABGEN	Same as above	Class password	Product# (HxxxxA+O3C) (HxxxxA+03D)	Customers with >20 systems;class partners
TEMPGEN	RCEs only	Temporary password (1-20 days)	RCO/SSO password; # of days	RCEs, CEs

Note VABGEN can also be run by CEs with 110 PCs for SuccessLine customers only.

Where and How to Get Passwords

North American Customers. If you are a licensed user (class or system-specific users), contact a call co-ordinator at the Response Center to obtain a password.

If you are an HP support person, you can obtain your password directly, by running the SSP110 program.

All Other Customers. If you are a licenced user (class or system-specific users), contact your Response Center Engineers to obtain a password.

If you are an HP support person, contact your Response Center Engineer to obtain a password.

License Installation Program (HP 9000 Series 800 Computer Systems Only)

Access to some tools is restricted to those users who have installed a system-specific/class license on their system: to install a system-specific/class license on your system, you must run the license installation program. If you feel you have a need to access the licensed support tool set, and do not have a system-specific/class license, please contact your Hewlett-Packard Account-assigned Customer Engineer.

Note A support temporary license is a special system-specific/class license, and should be installed in the same manner as the standard system-specific/class licenses.

To install a system-specific/class license, you must do the following:

1. Obtain a system-specific/class password from Hewlett-Packard (the procedures for doing so are provided with your system-specific/class license).

Note When the operating system is updated, the system-specific/class license must be re-installed. Also, since system-specific/class licenses expire, you must re-install your license, using a new password, after the expiration date.

2. Enter *suplicen* at the system prompt.

   The support license installation program (**suplicen**) employs a simple user interface. You run the installation program, enter your password when prompted to do so, and the installation program will display a message indicating whether or not the password you entered was accepted as valid.

The following is a sample run of the support license installation program on an HP-UX 9000 Series 800 system:

```
% suplicen
**********************************************************************
*****          Support License Installation Program            *****
*****                                                          *****
*****          (C) Copyright Hewlett-Packard Co 1992           *****
*****                 All Rights Reserved                      *****
*****                                                          *****
*****                   Version A.00.00                        *****
**********************************************************************

Please enter your Self-maintainer/class password (cr to exit) >
                user enters password here

YOUR SELF-MAINTAINER/CHANNEL PASSWORD HAS BEEN ACCEPTED. A
SELF-MAINTAINER/CHANNEL LICENSE IS NOW INSTALLED.

%
```

Note

When the system-specific/class password is entered, the characters will not be echoed back to you. System-specific/class passwords are typically 6 to 8 characters long. If you enter a single carriage return, the installation program will exit. If the password you enter is accepted, as in the example above, you will be able to access any licensed support tool. If you entered an invalid password, an error message will be displayed, and the installation program will exit. You will not be allowed access to any licensed support tool, until the system-specific/class license installation program accepts a password you enter as valid.

If you find that you are not able to install a support password using **suplicen**, you should check the following:

- Are SW_ID, SW_CAP, and H_VERSION set appropriately?
- Is /dev/diag/diag0 present?
- Is the date used by the system correct?

If all of the requirements above are met, you should have no problems in installing a support license.

Invoking the DUI

To enter the online diagnostics subsystem, one enters the command "SYSDIAG" at the system's prompt. Upon entering the DUI, the following banner will be displayed:

```
*****************************************************************************
******                                                                 ******
******                   ONLINE DIAGNOSTIC SYSTEM                      ******
******                                                                 ******
******        (C) Copyright Hewlett-Packard Co 1987,1988,1990          ******
******                   All Rights Reserved                           ******
******                                                                 ******
******                                                                 ******
******                   DUI Version XX.XX.XX                          ******
******                                                                 ******
******             Diagnostic Monitor Version XX.XX.XX                 ******
******                                                                 ******
*****************************************************************************

     Type "HELP" for assistance.


DUI >
```

The user is now free to enter diagnostic system commands. To exit the DUI, the user simply types EXIT.

Note

On B1 security systems, the user must be at "syslo", the lowest security level on the system, in order to run the DUI.

Version Identification

The DUI's version number is displayed in the banner. The format of the version is:

```
VERSION.UPDATE.FIX
```

Where *version* refers to major changes in large groups of system software, each *update* indicates a major change or addition to the Diagnostic User Interface, and a *fix* indicates bug fixes within the DUI.

Entering Commands

Replies and Responses

The user may be prompted from time to time for certain data. If one of several specific responses, such as **yes** or **no**, is required, the valid responses will be indicated in parenthesis. The default response - the assumed response if nothing is given but a carriage return - will be indicated in square brackets ([]). For example, if the user wants to exit and there are still running processes, he might be asked the question:

```
Do you wish to abort the currently running processes (Y/N) [N] ?
```

Valid responses include **Y**, **N**, **yes**, and **no**. Simply hitting the <RETURN> key selects the default response of **N**.

Continuation Lines

If a command is too long to fit on one line, the user may continue the command on subsequent lines by using the continuation character (\). This character causes the DUI to delay interpretation of the command. Usually, interpretation would begin immediately after the receipt of the carriage return. When the continuation character is given at the end of a command line (followed by a carriage return), the DUI prompt will appear on the next line - the user may then continue to type the command. A command may consist of numerous continuation lines provided that the total length of the text entered does not exceed 255 characters. For example:

```
DUI > run wizbang errcount=9 \
DUI >> ldev3  erronly
```

Note that the DUI prompt is slightly different to indicate that a continuation of the previous line is expected.

Command Comments

Command lines may include comments if desired. The beginning of a comment is indicated by the # character: the end-of-line automatically terminates the comment.

```
DUI > list              # This is a comment

DUI> # This is a
DUI> #multi-line comment
```

Installation Instructions

The DUI will be shipped to customers already installed in all releases; no user activity will be required. The set of diagnostic programs will also be installed in all releases prior to being shipped to customers. If a diagnostic program must be installed into the On-Line Diagnostic system at a customer's site the **install** command may be used at the DUI prompt.

Input and Output Files

The DUI works with *usefile* files, *infile* files, and *outfile* files. (See the USEFILE and OUTFILE commands and the INFILE and OUTFILE run command modifiers.) *Usefile* files and *infile* files are assumed to be comprised of lines 80 (ASCII) characters in length. *Outfile* files, which are made by the DUI, will also consist of 80 character length lines.

Usefiles, *infiles*, and *outfiles* may reside in any directory as long as the user has the appropriate capability to access them. The complete path name may be specified for all files and must be specified if the file resides somewhere other than in the directory the user is running the DUI from.

The form of complete path names differ among operating systems. For example, on MPE/iX a complete path name for **somefile** might be **somefile.mygroup.official** or just **somefile.mygroup**, while on HP-UX **/user/mydir/somefile** might be appropriate. It is assumed that the user is familiar enough with the file system organization in the operating system being used to be able to specify the appropriate path.

If the file name provided for an *outfile* already exists, it will not be overwritten, rather an error message will be printed.

Interactive Interface

The normal mode of operation of the DUI is interactive; however, interaction may be mimicked through a *usefile* (a file containing commands to be executed along with information the user would normally give interactively during the execution of those commands). Data which would usually be sent to a diagnostic by typing the information in at the user's terminal may also be entered automatically if an *infile* is specified. Conversely, information which would usually be sent directly from the diagnostic system to the user's terminal may be redirected by using an *outfile* or by giving the **hardcopy** command or modifier.

Usefile, *infile*, *outfile*, and **hardcopy** are more fully explained in their own sections of this document.

Programmatic Interface

A simple programmatic interface to the DUI exists. A program may invoke the DUI by using whatever construct is provided in the operating system being used to execute a command interpreter or shell command from within a program. The command to be executed is SYSDIAG. The program may send a one line command, in the form of an information string, to the DUI.

The one line may be any command the DUI would normally accept, including the **usefile** command. The DUI will terminate immediately after executing the one line command.

For example, on MPE/iX the intrinsic **create** or **createprocess** could be used while on HP-UX one of the **exec** system calls would be used.

Interrupts

The effects of typing a user interrupt while in the DUI depend upon what action is taking place. Interrupts will only affect diagnostic programs and processes which are running in the foreground. Background processes cannot be interrupted.

- If no program is running in the foreground and no USEFILE file is being used
  - "**** INTERRUPT ***" will be printed on the standard output device
  - The DUI prompt will reappear
  - Nothing else will be done.
- If executing a USEFILE
  - "**** INTERRUPT ***" will be printed on the standard output device
  - Reading of the *usefile* is stopped and the *usefile* is closed. All *usefiles* nested with the current *usefile* are also closed.
  - If a program is running in the foreground it will either receive a message that an interrupt has occurred or will be suspended as explained below.
- If a program has requested control of its own interrupts
  - The program will be informed of the interrupt. What then occurs is program dependent.

- If a program is running in the foreground and has not requested control of its own interrupts
  - "*** INTERRUPT ***" will be printed on the standard output device
  - The program will be SUSPENDED.
  - The DUI prompt will reappear

The control characters used to generate a user interrupt are operating system dependent. On MPE/iX CTRL-Y works; on HP-UX CTRL-C.

Security

Many diagnostic tests have the indirect effect of corrupting data (e.g., selftests which write into onboard registers). To ensure system and user data integrity, the online diagnostics subsystem has its own security mechanisms which augment the normal operating system security. The online diagnostics subsystem also uses its own security mechanisms to ensure diagnostic system integrity.

Commands given to the DUI will not be executed if the caller has too low a security level. Security level is sometimes called capability level - the two are synonymous.

The online diagnostics subsystem maps the user's operating system security into one of four diagnostic system security levels:

1. diagnostic security level 0
2. diagnostic security level 1
3. diagnostic security level 2
4. diagnostic security level 3

Diagnostic security level 0 is the highest; level 3 is the lowest.

The correspondence between the user's operating system security level and the user's diagnostic security level is operating system dependent. This is necessary because the different operating systems vary in the way different classes of security are defined and in the kinds of security granted different users. However, some general statements may be made as to the diagnostic security levels various users would be given.

An HP-UX superuser (root) or an MPE/iX system manager would be assigned the highest diagnostic security level, level 0, and could perform any action the diagnostic system was capable of performing. A user who had only the lowest level of security available on an operating system would be assigned the lowest diagnostic security level, level 3, and could perform only those actions which were non-destructive and which did not require exclusive access to a device. This would include, in most cases, such actions as running diagnostic sections which copied and decoded the *identify* block from a hardware board.

Users whose operating system security fell somewhere between the lowest and highest would be assigned diagnostic security level 1 or level 2 and could perform such actions as sending loopbacks to devices. But again, the diagnostic security level assigned to a user and hence the actual tests and actions which a particular user could perform are operating system dependent.

Users on MPE/iX with operating system capability **sm** are given diagnostic security level 0. Those with `di` capability are given diagnostic security level 1. Those with op capability are given diagnostic security level 2. All other users are assigned diagnostic security level 3.

An HP-UX user is assigned a diagnostic security level based on whether or not the user has an entry in the diagnostic SECURITY file. Each entry in this file consists of a user's login name, along with a number. Those users whose names are followed by the number "0", such as root, are given diagnostic security level 0. Those users whose names are followed by "1" are given diagnostic security level 1. Those whose names are followed by "2" or "3" are given diagnostic security level 2 or 3, respectively. Thus, a SECURITY file might look like this:

```
root:0
jdoe:2
jroe:0
tsmith:1
```

If the SECURITY file cannot be found, or the user's login name cannot be found in it, the user is assigned diagnostic security level 3.

The HP-UX diagnostic SECURITY file is maintained by the system administrator, and can only be modified by a superuser. The SECURITY file is located in the same directory as the other major diagnostic files, normally **/usr/diag/bin**. Please see the system administrator if a new entry is needed in this file.

Every action which might be performed through the online diagnostics subsystem is associated with one of four diagnostic security states.

These states are:

1. non-destructive/non-exclusive
2. non-destructive/exclusive
3. destructive/non-exclusive
4. destructive/exclusive

Non-destructive and *destructive* (the two possible test modes) indicate the possibility that an action could destroy data. *Non-exclusive* and *exclusive* (the two possible access modes) refer to whether or not a device must be accessed exclusively (locking all other processes out during the time of use) to perform an action.

Many actions do not directly involve devices so, for those actions, the access mode is moot. For example, purging a diagnostic using the PURGE command does not directly involve a device but has a drastic effect on the diagnostic system; the DUI would only allow a user whose diagnostic system security level was such that he could run destructive tests to execute the PURGE command.

The online diagnostics subsystem checks the user's diagnostic security level, whether the device being tested is being accessed exclusively or non-exclusively (access mode), and the level of potential destructiveness (test mode) of the requested tests or the given commands before allowing any particular set of tests to run or commands to execute. The state of the operating system also affects which commands and tests a user can run through the DUI.

For example, a user whose diagnostic security level was such that he would normally not be allowed to perform a *destructive* action on a *non-exclusive* device would be allowed to do so if the operating system had been booted in single-user mode.

If a user attempts to run a test or execute a command which requires a higher security level than he possesses, a message will be printed explaining why the action could not be performed.

COMMANDS - GENERAL INFORMATION

Notation and Special Symbols

The following notation is used in the command and command modifier syntax diagrams. The notation is also used in messages printed in response to the **help** command.

`[]` An element inside brackets is *optional*. Several elements stacked inside brackets means the user may select any one or none of these elements. For example:

```
[A]
[B]
[C]  User may select A or B or C or none.
```

When brackets are nested, parameters in inner brackets may be specified only if parameters in outer brackets are specified.

For example:
[parm1 [parm2 [parm3]]]

may be entered as
parm1 parm2 parm3
or
parm1
or
parm1 parm2

Optional parameters which are not positional are shown as follows:

[parm1][parm2]

`[,...]`
or
`[...]` Means that the immediately preceding item in the syntax diagram may be repeated any number of times. The [,...] form means that each instance of the repeated item must be preceded by a comma.

`{}` An element inside curly braces is *required*. When several elements are stacked within braces in a syntax statement, the user must select one of those elements. For example:

```
{A}
{B}
{C}  User must select either A or B or C.
```

| | Vertical parallel lines indicate that any or none of the options may be used in any sequence. However, the elements may not be repeated. For example:

```
|A|
|B|
|C|  The user may choose A, B, and C; or C and A,
     or B alone, etc.
```

UPPERCASE   Represent literals which are to be entered exactly as shown except that they may be entered in lower case. Also, if an abbreviation or alternate token is listed for the item, a substitution is allowed. Lastly, only the first *n* characters which establish uniquely what the literal is need be entered.

Special Character Literals

The special characters + - / ( ) " = are literals to be entered exactly as shown in the syntax diagrams.

italics   Items printed in *italics* are to be replaced with user supplied information.

Delimiters and Abbreviations

All keywords and options may be abbreviated to the shortest number of characters which make the token unique. Delimiters for keywords and options include spaces and/or semicolons (;). Other delimiters may be specified in the descriptions and syntax diagrams for individual commands and command modifiers.

UPPER and lower case command lines are equivalent.

User Set Default Values

The user may specify default values which differ from the normal diagnostic system default values for command modifiers using the **set** command.

The user may set a "universal" default value for a modifier - that is, a value which will have effect at all times - or a default value which will take effect only when a particular diagnostic is run. In either case, if another value for the modifier is given in a **run** command, that value will have effect only until the diagnostic named in the **run** command completes execution.

Modifier default values which have been changed, either universally or for individual diagnostics, using the **set** command can be reset to their diagnostic system default values using the **default** command.

For more information, please see the **set** command and the **default** command descriptions in this document.

Designating Devices to be Tested

Usually, when a diagnostic program is invoked the device to be tested must be named. This can be done in one of two ways: either by using the logical name of the device or giving the physical path address of the device. That is, one of the two command modifiers **ldev** or **pdev** must be given with the **run** command, or must be set using the **set** command.

The form these strings take varies among the operating systems and even from one HPPA machine to another.

For example, under the MPE/iX operating system the logical name (**ldev**) of a device is a number (e.g., 3) while in the HP-UX operating system the logical name of a device is that of a special device file (e.g., dsk/c0d0).

The physical path address (**pdev**) of a device is a series of numbers separated by various punctuation marks. Each number in the series corresponds to a physical connection along the electrical pathway to the device. For example, an HPIB device adapter might have an address of 4.2 on an HP3000 series 930, but an address of 2/8.0 on an HP9000 series 850.

It is assumed that the user of the diagnostic system knows enough about the operating system and machine being worked with to determine the correct **ldev** or **pdev** for the device being tested. A utility program, SYSMAP, is provided in some diagnostic installations and may be run through the DUI (SYSMAP may NOT be available in all installations) to help determine the desired **ldev** or **pdev**.

Running Multiple Diagnostics

Several diagnostic programs may be invoked simultaneously using the **run** command. To do this, the user simply names all of the diagnostics to be run along with the command modifiers for each in a **run** command.

Examples:

```
DUI> run xdiag ldev 0 loop 5 erronly ydiag pdev 4.3.2 \
DUI>> errpause background outfile yout

DUI> qdiag ldev 3 infile qinf & rdiag ldev 0 \
DUI>> sc 4/5(20,23)
```

All but one of the programs must be run in the background; only one program may run in the foreground at any one time.

It is suggested that the output of programs run in the background be redirected to an *outfile*. The DUI will not impose any order on the messages received from multiple diagnostic programs but will output each message as it is received.

When multiple diagnostic programs are invoked, the DUI will do all the internal checks and initial set up it would usually do for each of the diagnostics. THEN it will launch the diagnostics one immediately after the other using the mechanisms provided by the operating system. The DUI will not wait for one of the diagnostics to complete before launching the next.

The number of diagnostics which may be run simultaneously is dependent on the number of processes the operating system will allow any one user to run simultaneously - the diagnostic system itself does not impose a limit.

Command Summary

The following is a list and brief description of each of the commands available in the DUI.

Command Name	Description
ABORT	Terminates active diagnostic program or utility.
CI	Provides access to the operating system command interpreter.
CODETEST	For HP internal use only.
DEFAULT	Resets command modifiers to their original diagnostic system default values.
DIAGSYSTEM	Used to access and manipulate internal diagnostic system processes which are usually not accessible to the user.
DO	Allows the user to re-execute any command in the command line history stack. It also permits the user to edit the command before re-executing it.
EXIT	Terminates the DUI and returns control to the operating system.
FOREGROUND	Moves a diagnostic which has been running in the background into the foreground.
HARDCOPY	Causes all terminal input/output to be echoed to the system printer.
HELP	Accesses HELP facility for information about the DUI and its commands or for information about any of the diagnostic programs.
INSTALL	Allows the user to install diagnostic programs.
LIST	Provides information about any or all of the programs in the diagnostic system.
LISTREDO	Displays the user's command history stack.
MODE	Displays and/or alters the current operating system mode.
MODIFY	Allows a user to change information about a diagnostic program without needing to reinstall the diagnostic.
OUTFILE	Causes all diagnostic system input and output to be written into the specified file.
PURGE	Removes programs from the diagnostic library.
REDO	Allows user to display (for command editing) and re-execute any command in the command line history stack.
REDOLOAD	Replaces the user's command history stack with the history stack which was saved by the command REDOSAVE.
REDOSAVE	Causes all or part of the user's command history stack to be saved into a file.

REDOSIZE	Allows the user to set the maximum number of commands which will be saved in the user's command history stack.
REPLY	Used to send a reply to the prompt of a diagnostic program which is running in the background.
RESUME	Resumes processing of a suspended program.
RUN	Loads and executes a specified program.
SETVAR	Allows the user to explicitly set the values of environmental variables which control various features of the user interface.
SET	Allows the user to explicitly reset the system default values for modifiers which may be given on a **run** command line.
SHOWACTIVE	Lists programs which are currently active for the user.
SHOWDEFAULT	Causes a list of command modifiers and their default values to be displayed.
SHOWPARMS	Causes a list of user settable modifiers and their current values to be displayed.
SHOWSTATE	Will cause the current system mode (single-user or multi-user) and the user's security capability to be displayed.
SUSPEND	Suspends processing of a specified program.
UNLOCK	Releases specified device from lock status.
USEFILE	Causes input to the diagnostic system to be input from the specified file rather than from a user's terminal.

The following pages provide a detailed description of each of the available DUI commands. The description includes information about syntax, options available for each command, limitations of each command, and examples.

ABORT

The abort command terminates a program.

The chosen diagnostic program will be aborted regardless of what it is doing. The diagnostic system may perform clean up actions on any devices allocated to the program but this is not guaranteed. If more than one program is running or in a suspended state, the *program-id* modifier or ALL modifier should be given. If only one program is running or suspended when the `ABORT` command is given no modifier is necessary.

Syntax:

```
Abort [ALL                 ] [UNCONDITIONAL]
      [program-id [[,]... ]] [UNCONDITIONAL]
```

Modifiers:

program-id — The process identifier number of the diagnostic program to be aborted. If there is more than one active program and the *program-id* is not specified, a list of *program-ids* from which to choose will appear.

ALL — Abort all programs running under the diagnostic system. This will cause all programs running under the DUI through which the abort command was received to terminate.

UNCONDITIONAL — Has no effect in online diagnostics subsystem; all aborts are unconditional.

Command Examples:

```
DUI> abort 23

DUI> abort

DUI> abort 4 32
```

CI

CI (command interpreter) invokes the system's command interpreter or shell so that one or more operating system commands may be executed.

If the CI command is given without a modifier, the command interpreter prompt, which is system specific, will appear. Command interpreter commands can then be given until the user specifically exits back to the DUI. The command used to exit back to the DUI is system specific - **exit** on MPE/iX and on HP-UX systems.

If the CI command is given with a *command* that one command will be executed and the DUI prompt will reappear.

Syntax:

CI [*command*]

Abbreviations and Alternative Tokens:

!

:

Modifiers:

command	The command (and its arguments) which is to be executed by the system's command interpreter or shell.

Command Examples:

```
DUI> ci
%                          {the command interpreter's prompt}

DUI> : listf

DUI> !ls
```

Limitations:

Warning

If the operating system command to exit a session or to begin a new session is given (e.g., logout on HP-UX or bye on MPE/iX), the DUI will be terminated. No clean up will be done. This is especially dangerous if the system was placed into single-user mode through the DUI (see MODE command). The system may need to be rebooted before anyone, including the console operator, will be able to log on.

CODETEST

This command is provided for Hewlett-Packard internal use only.

DEFAULT

DEFAULT resets command modifiers to their original diagnostic system default values. (The modifiers could have had their values reset by the **set** command.)

DEFAULT will have no effect if it is used with a modifier which still has its original diagnostic system default value. No error message or warning will be printed.

If the command is given without an argument the effect is the same as if the user typed DEFAULT ALL.

Syntax:

```
DEFAULT  [command modifier [[,]...]]
         [ALL                      ]
```

Modifiers:

command modifier One of the following

```
BACKGROUND
DEBUG
ERRCOUNT
ERRONLY
ERRPAUSE
ERRPRINT
HARDCOPY
INFILE
LDEV
LOOP
OUTFILE
PDEV
SECTIONS
TRACE
Diagnostic Specific Parameters
```

Please note that this list includes all run command modifiers except SECURITY.

The "Diagnostic Specific Parameters" can be reset to null - the diagnostic system default - by giving a pair of double quotes ("") as a modifier to DEFAULT.

ALL Resets all modifiers, including the "Diagnostic Specific Parameters" modifier to their diagnostic system default values.

Command Examples:

```
DUI> default loop,"",trace,hardcopy

DUI> default all
```

Related Commands:

SET
SHOWDEFAULT
SHOWPARMS

DIAGSYSTEM

DIAGSYSTEM is used to access and manipulate internal diagnostic system processes which are usually not accessible to the user. DIAGSYSTEM may be used to get a listing of running diagnostic system processes along with their program identifiers; abort diagnostic system processes: and launch diagnostic system processes. These processes run outside of the user interface and their execution cannot be altered by the user. For example, some diagnostic system software is sold by HP as an add-on product - DIAGSYSTEM allows this software to be added to a customer's system without needing to do an **xinstall** and **reboot**. This command also obviates the need to reboot the system to launch diagnostic system processes. Examples of such processes would be diagnostic logging processes and diagnostic statistical analysis processes.

Syntax:

DIAGSYSTEM {SHOWACTIVE }
{ABORT *prog_id*}
{RUN *prog_name*}

Abbreviatons and Alternate Tokens:

DS

Modifiers:

SHOWACTIVE	Display a list of running diagnostic system processes along with their program identifiers.
ABORT *prog_id*	Terminate the diagnostic system process specified by the given program identifier.
RUN *prog_name*	Launch the diagnostic system process specified by *prog_name*.

Command Examples:

```
DUI> diagsystem showactive

DUI> ds run memlogp

DUI> diagsystem abort 33
```

Limitations:

Only a user with the highest diagnostic security level (diagnostic security level 0) will be allowed to abort a diagnostic system process.

DO

DO allows the user to re-execute any command in the command line history stack. It also permits the user to edit the command before re-executing it. The (edited) command is executed immediately after the carriage-return; no interactive editing may occur. (The REDO command must be used if interactive editing is desired.)

Syntax:

```
DO [[CMD=]command-id] [, edit-directives      ]
                      [;EDIT= edit-directives]
```

Modifiers:

command-id — Specifies the command to re-execute. The command may be specified by its relative or absolute order in the command line history stack, or by name (as a string) in whole or in part. The default *command-id* is -1, the most recent command. An error is detected if the *command-id* does not exist in the command line history stack.

COMMAND-ID	EXECUTES
(omitted)	The most recent command (same as DO -1).
-n	nth command before the most recent one. N is a number in the command line stack relative to the most recent command, which is -1.
m	Command number m in the command line stack. The number m is absolute (not relative).
string	The most recent command beginning with this string.

edit-directives — String specifying the changes to be made in the command represented by the *command-id* before its re-execution. If no *edit-directives* are given, the command is re-executed immediately - no editing is performed.

Note

Both *command-id* and *edit-directives* must be surrounded by quote marks ("or ') if they contain any delimiters such as: . ; " ' [ ] = or a space.

When an editing command such as replace, insert, delete, or change is to take effect anywhere other than in column 1 of the original command, the position (in the original command string) where the edit should begin can be designated by preceding the edit command with spaces.

Example:

: diuspend   *mistyped command*
: do ,drs   *delete the 'd' in col. 1 and replace the 'i' with an 's'*
: suspend   *result of edit*

: susp1lnd   *mistyped command*
: do ,"   ddie"   *delete the fifth and sixth characters and insert an 'e'*
: suspend   *result of edit*

The editing directives which may be used as *edit-directives* are:

DIRECTIVE	EFFECT
i	INSERT. If text follows the i, the text following i is inserted in the current line at the position after the i.
r	REPLACE. If text follows the r, the text following r replaces the same number of characters in the current line, beginning at the position of r.
d	DELETE. Deletes a character from the current line for each d specified in the edit line. Note that "d d" does not specify a range, but simply deletes one character from the position above each d. Multiple d's may be followed by an insert or replace operation.
d>	DELETE. Deletes to the end of the current line from the position specified by d>. It may be followed by an INSERT or REPLACE operation.
>	APPEND. > followed by text appends the text to the end of the current line. If > is positioned beyond the end of the current line, then a replacement is performed instead.
>d	DELETE. Deletes from the end of the current line, right-to-left. Multiple d's may be specified after >, as well as INSERT and REPLACE strings.

>r REPLACE. Replaces characters at the end of the command line. The replacement is done so that the last (rightmost) character of the replacement string is at the end of the line.

c CHANGE. Changes all occurrences of one string into another in the current line when the searched for string and the replacement string are properly delimited. A proper delimiter is a non-alphabetic character: ', ", /, etc.

The substitution is specified as:

c<delim>*search-string*<delim>
[*replace-string*[<delim>]]

Omitting the *replace-string* causes occurrences of the *search-string* to be deleted.

other Simple replacement. Any other character (not i, r, d, d>, >, >d, c, or u) causes that character to be replaced in the current line at the position indicated by the character. In fact, simple replacement also occurs for the editing characters i, r, c, or > if they are not followed by text; or if > appears at or beyond the current end of line.

Command Examples:

EDITING SAMPLES

Practical uses of the editing commands listed above are shown here:

EDIT	ACTION
rxyz	Replaces the current text with xyz starting at the position of r.
xyz	Replaces the current text with xyz starting at the position of x.
ixyz	Inserts xyz into the current line, starting at the position immediately before the i.
ddd	Deletes three characters, one above each d.

"d xyz"	Deletes a single character above the d, skips one space, then replaces the current text with xyz starting at the position of x.
ddixyz	Deletes two characters, then inserts xyz in the current line in the position before the i.
'd d'	Deletes one character above the first d, skips two spaces and deletes a second character above the second d. It does not delete a range of characters.
'd d>xyz'	Deletes a single character above the first d, skips two spaces and deletes to the end of the line beginning at the second d, and then appends xyz to the end of line.
>xyz	Appends xyz to the end of the current line.
>ddxyz	Deletes the last two characters from the end of the current line and then appends xyz to the end of the line.
>rxyz	Replaces the last three characters in the current line with xyz.
>ixyz	Appends xyz to the end of the line. In this case, the i command is superfluous, because > accomplishes the same result. Using >xyz would be sufficient.
c/ab/def	Changes all occurrences of ab to def, starting at c.
c"ab"	Deletes all occurrences of "ab" starting at c.
cxyz	Replace the current text with cxyz, starting at c. Because delimiters have not been specified (as they were in the previous two examples), this is a simple replacement.

EXAMPLES

DO pas	Re-executes the the most recent command beginning with the string pas.
DO 10	Re-executes command number 10 (absolute) on the command history stack.

`DO -2`	Re-executes the second-to-last command on the stack (one command before the most recent).
`DO -2, c/5a/5b`	Change all occurrences of 5a to 5b in the command preceding the most recent one before re-executing it. The default is -1.
`do ,c/5a/5b`	Change all occurrences of "5a" to "5b" in the most recent command before re-executing it.
`DO run, ">;debug"`	Append ;debug to the the most recent :RUN command and then re-execute it.
`DO 'RUN MYP', '>;LIB=G'`	Find the most recent command beginning with RUN MYP and append ;LIB=G before re-executing it.

Related Commands:

```
REDO
LISTREDO
REDOLOAD
REDOSAVE
REDOSIZE
```

Limitations:

- DO is based on an MPE/iX command.

EXIT

EXIT causes the DUI to terminate. The system prompt will then appear. If any diagnostic programs are running or suspended when the command is received the user will be asked if he wishes to abort them - if the answer is no the EXIT will not be processed; the user will remain in the DUI.

An exception to this will occur if either the QUIET or the UNCONDITIONAL modifiers are used. If either or both of these modifiers are used all diagnostic programs existing under the DUI (suspended, running in background, etc.) will be aborted automatically - the user will not be queried.

Syntax:

```
EXIT |QUIET        |
     |UNCONDITIONAL|
```

Abbreviations and Alternate Tokens:

```
E
```

Modifiers:

```
QUIET            Causes all diagnostics running under the DUI
                 to be aborted before the DUI is exited. The
                 user is not queried to confirm that a process
                 should be aborted.

UNCONDITIONAL    Has the same effect as the QUIET modifier
                 except that all programs running under the
                 DUI will be aborted even if it means leaving
                 the system in an unusual state; normal
                 clean up will not necessarily be done. For
                 example: devices under test might remain in
                 a locked state; tables within the diagnostic
                 system might not be updated to reflect the
                 actual state of the system.
```

Command Examples:

```
DUI> exit

DUI> exit unconditional

DUI> exit quiet

DUI> exit unconditional quiet
```

naming both modifiers is not an error although only the "unconditional" will really have an effect

Limitations:

- The UNCONDITIONAL modifier may leave the diagnostic system and the I/O system in an unknown state: there will not necessarily be any graceful way to recover from this. The effects of this modifier may not be consistent across different operating systems nor even consistent from one session to another on the same machine.

FOREGROUND

The `FOREGROUND` command moves a diagnostic which has been running in the background to the foreground. If a program identifier (a process id number) is not specified and only one diagnostic program is running in the background, that program will automatically be placed in the foreground.

If more than one process is running in the background, but the user does not specify which is to be brought to the foreground, the user will be given a list of the processes running in the background from which to choose.

Syntax:

FOREGROUND [*program-id*]

Abbreviations and Alternate Tokens:

FG

Modifiers:

program-id The process identifier of the diagnostic program which is to be brought into the foreground.

Command Examples:

```
DUI> fg

DUI> foreground 17
```

Related Commands:

RUN (with BACKGROUND modifier)

Limitations:

- Only one diagnostic program may run in the foreground at any time.
- Once a program is running in the foreground it cannot be placed in the background again: it must run to completion in the foreground.

HARDCOPY

The HARDCOPY command causes all further input to and output from the diagnostic system to be printed by a hardcopy device. Input and output will also continue to appear on the standard output device.

This will continue until HARDCOPY is turned off or the diagnostic system is exited.

Syntax:

```
HARDCOPY [LDEV [=] logical device name [ENV [=] environment]]
         [PDEV [=] physical path       [ENV [=] environment]]
         [OFF                                              ]
         [ON                                               ]
```

Abbreviations and Alternate Tokens:

HC

Modifiers:

LDEV *logical device name*

The logical name of the hardcopy device which is to be used.

PDEV *physical path* — The physical path address of the hardcopy device to be used

ENV *environment* — Information which may be used by the hardcopy device to control printing in some way. What this information is and how it must be stated varies from hardcopy device to hardcopy device and from operating system to to operating system. This argument is provided as a convenience for users who wish to control their printing environment and who are knowledgeable about the "environments" used by the chosen hardcopy device.

The diagnostic system will use some printing environment by default if the user does not specify one by using the ENV argument.

OFF — Cease to echo input and output to the hardcopy device.

Command Examples:

```
DUI> hc

DUI> hardcopy ldev 23
```

Related Commands:

```
OUTFILE
RUN (with OUTFILE modifier)
```

Limitations:

- Output may not go to the printer immediately; it may be spooled first.

HELP

The HELP command is used to get information about the diagnostic system. Using the HELP command without stating a topic will cause general information about the on-Line diagnostic subsystem to appear. If a topic is specified, information about that topic will appear. HELP messages are available for every diagnostic program and every section and step in every diagnostic. HELP messages also exist for every command listed in this document.

Syntax:

HELP [HELP]
[*command* [SYNTAX]]
[*run command modifier* [SYNTAX]]
[MNEMONICS]
[*program name*]
[*program name* SECTIONS]
[*program name* SC]
[*program name* SECTION *n*]
[*program name* SC *n*]
[*program name* (*n*)]
[LDEV [=] *logical device*]
[LDEV [=] *logical device* ACCESS]
[LDEV [=] *logical device* ID]
[PDEV [=] *physical path*]
[PDEV [=] *physical path* ACCESS]
[PDEV [=] *physical path* ID]

Abbreviations and Alternate Tokens:

H
?

Modifiers:

omitted General information about the online diagnostics subsystem will be printed.

HELP HELP used as a modifier to the HELP command will cause a complete list of the topics about which HELP can give information to be printed.

command Causes a brief description of the purpose of the *command* to appear along with a syntax diagram. A help message exits for every command in this document.

command SYNTAX
Causes only the command syntax to appear, not a description of the command.

run command modifier

Causes a brief description of the purpose of the *run command modifier* to appear along with a syntax diagram. A help message exits for every run command modifier in this document.

run command modifier SYNTAX

Causes only the command syntax to appear, not a description of the run command modifier.

MNEMONICS

Causes a list of all currently recognized mnemonics to be output.

program-name

Causes a general message about the diagnostic to appear. Although the specific contents of this message will vary from diagnostic to diagnostic, typically, the message will state what the diagnostic tests and what the diagnostic is capable of doing.

program-name SECTIONS
program-name SC

Causes a list of all the sections available in the named diagnostic to appear along with a brief description of what each does. A list of mnemonics which may be used in place of section numbers will also appear. Each mnemonic will be annotated with a list of the section numbers it replaces.

program-name SECTIONS *n*
program-name SC *n*

Causes a detailed description of section *n* of the named diagnostic to appear. If the section is divided into steps, a list of these will appear. If any of the steps may be replaced with mnemonics, a list of those mnemonics annotated with the steps they may replace will also appear.

program-name (*n*)

Causes a detailed description of step *n* of the named diagnostic to appear.

LDEV *logical device*
PDEV *physical path*

> Results in a message stating the access mode (exclusive or non-exclusive) and the test mode (destructive or non-destructive) of the device. The user will be told what must be done to get the diagnostic security level necessary to access the device. The product name of the device (e.g., HP1234B) will also be given.

LDEV *logical device* ACCESS
PDEV *physical path* ACCESS

> The access mode (exclusive or non-exclusive) and the test mode (destructive or non-destructive) of the device will appear. A message will also be output stating what the user must do to get the diagnostic security level needed to access the device.

LDEV *logical device* ID
PDEV *physical path* ID

> The product name of the device will appear. This will be some number such as HP1234A.

Command Examples:

```
DUI> h

DUI> ? foodiag

DUI> help foodiag sections

DUI> h foodiag sc 5

DUI> h foodiag (155)

DUI> help pdev 4/2.3

DUI> ? mnemonics

DUI> ? h

DUI> ? resume

DUI> h h syntax
```

INSTALL

The `INSTALL` command is used to add programs to the diagnostic system.

The `INSTALL` command causes the `DUI` to update diagnostic system files with information about the program to be installed. It also places the program's message catalog and the executable program file in their correct places in the diagnostic system. Special files (such as downloadable code files) needed by the program are also taken into the diagnostic system using the `INSTALL` command.

The user is expected to have ready an executable copy of the program and a GENCAT formatted copy of the message catalog. The user is also expected to have ready any special files peculiar to the program. These files may be formatted in any way needed by the program.

Information about the program, which is used each time the program is run, is gathered from the program's message catalog when the `INSTALL` command is given.

The `INSTALL` command is used only to add new programs to the diagnostic system: the user will not be allowed to install (add) a program which already exists in the diagnostic system. Instead, the user may

1. revise information about an already installed program by using the `MODIFY` command, or
2. reinstall a program by first removing it from the diagnostic system using the `PURGE` command, or
3. install the program under a different name.

Information about the diagnostic is gathered from messages in the installation set of the diagnostic's message catalog. The content of these messages is as follows:

```
1 - Program version number in the v.uu.ff format
    (e.g., A.01.09).

2 - Catalog version number in the v.uu.ff format
    (e.g., A.01.09).

3 - Type of program being installed. Options are:

              DIAGNOSTIC

              UTILITY

              EXERCISER

              VERIFIER
```

```
4 - Devices diagnosed by the program.

    ID          [AUTO] [DECODE]

     :            :       :        %

     :            :       :        %

     :            :       :        %

{Where:                                               }
{ ID       = Product number for diagnosed device      }
{            (e.g., HP7978B)                          }
{ AUTO     = If specified, this indicates that the    }
{            program serves as the designated auto-   }
{            diagnostic for this device.              }
{ DECODE   = If specified, this indicates that the    }
{            program serves as the designated hardware }
{            status decoder for this device.          }
```

Please note the percent signs (%) in the above example. These tell the message formatter, GENCAT, that the following line is part of the message. The percent signs **must** appear at the end of every line except the last in message 4.

If any of the above messages contains an invalid value, an error message descriptive of the exact problem will be issued and the installation will be aborted. It will then be up to the user to fix the message catalog and reinstall.

The user will be prompted for four additional pieces of information which are needed to complete the installation process. These are:

- The location (file path) of the diagnostic system into which the installation is to be done. This allows experimental/working diagnostic subsystems to be built and maintained for development and integration purposes. The default will always be the location of the diagnostic system originally issued with the operating system.
- The name of the file containing the executable program. This name must be specified as fully as is necessary to locate it in the file system.
- The names of the files containing the formatted message catalogs. These names must be specified as fully as is necessary to find them in the file system. The message catalog files must be the formatted output of the Native Language Support catalog formatter, GENCAT.

  The user will also be asked to specify the language of each message catalog. The language of the message catalog must be specified in the same language in which the DUI is currently printing messages. For example, if the DUI is currently using English and a German language catalog is being installed, the language of the catalog would be specified as being "German." If the DUI is currently using German, the language of a German language catalog would be specified as "Deutsche."

Caution

MOST IMPORTANTLY, THE LANGUAGE OF THE FIRST MESSAGE CATALOG INSTALLED FOR EACH DIAGNOSTIC PROGRAM MUST BE THE SAME LANGUAGE THE DUI IS CURRENTLY USING TO PRINT MESSAGES.

- The DUI reads the installation information for a diagnostic program from the installation set of the first message catalog installed for that program. If this information is in a different language than that of the DUI, the DUI will not be able to process it and the installation will fail.
- The names of any additional files used by the program. Such files might, for example, contain downloadable code or environmental scripts used by the program.

The information gathered from the message catalog will be displayed (in the same format as for the LIST command). This assumes that no error occurred while getting the information from the message catalog.

If the installation is successful, a message indicating so will be generated and the user will be returned to the DUI prompt. If the installation fails, messages explaining the exact nature of the problem will be displayed before the DUI prompt reappears.

Syntax:

```
INSTALL program-name
```

Abbreviations and Alternate Tokens:

none

Modifiers:

program-name	The name by which the program will be referred when issuing the RUN command.

Data Prompts:

```
Specify the file path of the diagnostic system into which the
program will be installed:
FILE PATH [<cr>= cccccccccccccc] >
```

{*Where* cccccccccccccc *will be the file path to the group or directory where the "normal" or "default" diagnostic system resides*}

```
Specify name (qualify as necessary) of executable program file:
FILE NAME [<cr>= cancel install] >
```

```
Specify name (qualify as necessary) and language of message
 catalog file:
FILE NAME (current language) [<cr>= cancel install] >
FILE NAME [<cr>= no more message catalogs] >
   LANGUAGE [<cr>= cancel this file] >
```

```
FILE NAME [<cr>= no more message catalogs]  >
   LANGUAGE [<cr>= cancel this file]  >
     :
     :

Specify name (qualify as necessary) of another associated file:
FILE NAME [<cr>= no more files]  >
FILE NAME [<cr>= no more files]  >
    :           :
    :           :
```

Output:

The following program is being added to *name of diagnostic subsystem*

Name	Program Version	Program Type	Catalog Languages	Devices	Associated Files
========	========	=======	===========	============	==========
cccccccc	*c.cc.cc*	*cccc*	*cccccccccc*	*ccccccc* ad*	*cccccccc*
				ccccccc	*cccccccc*
			:	:	:
			:	:	:
			:	:	:

```
* a  program is the auto-diagnostic for this device
  d  program is the hardware status decoder for this device
```

Command Examples:

```
DUI> install foodiag
```

Related Commands:

```
MODIFY
PURGE
```

Limitations:

- The installation will be rejected if the named program already exists in the diagnostic system.

LIST

The LIST command causes a list of programs installed in the diagnostic system to appear. The modifiers used determine which programs are listed and what additional information is given for each.

Syntax:

```
LIST [program-name [[,]... ]] [LONG ]
                              [SHORT]
     [PRODUCT product-name   ] [LONG ]
                              [SHORT]
     [TYPE {DIAGNOSTIC}      ] [LONG ]
           {EXERCISER}        [SHORT]
           {VERIFIER}
           {UTILITY}
```

Abbreviations and Alternate Tokens:

L

Modifiers:

omitted — A list of all programs installed in the diagnostic system will be printed.

LONG — The *program-name* information for all installed diagnostic programs is printed. (See *program-name* LONG modifier, below).

SHORT — A list of all programs installed in the diagnostic system will be printed. (Same effect as giving the LIST command without any modifiers).

program-name
program-name SHORT
The program name is printed.

program-name LONG
The program name is printed along with its

* executable program (code) version number
* message catalog languages
* type (diagnostic, exerciser, verifier, or utility)
* auto-diagnostic responsibilities
  (i.e., is it an auto-diagnostic)
* decoding responsibilities
  (i.e., may it be used to decode hardware status)
* products which the program tests
* associated files (such as downloadable code files

used by the program)

PRODUCT *product-name*
PRODUCT *product-name* SHORT
The programs which run with the specified product are listed.

PRODUCT *product-name* LONG
The programs which run with the specified product are listed along with complete information about each program.
(See *program-name* LONG modifier, above).

TYPE DIAGNOSTIC
TYPE DIAGNOSTIC SHORT
Lists the 'diagnostic' programs.

TYPE DIAGNOSTIC LONG
Lists the 'diagnostic' programs with complete program information.
(See *program-name* LONG modifier, above).

TYPE EXERCISER
TYPE EXERCISER SHORT
Lists the 'exerciser' programs.

TYPE EXERCISER LONG
Lists the 'exerciser' programs with complete program information.
(See *program-name* LONG modifier, above).

TYPE VERIFIER
TYPE VERIFIER SHORT
Lists the 'verifier' programs.

TYPE VERIFIER LONG
Lists the 'verifier' programs with complete program information.
(See *program-name* LONG modifier, above).

TYPE UTILITY
TYPE UTILITY SHORT
Lists the 'utility' programs.

TYPE UTILITY LONG
Lists the 'utility' programs with complete program information.

(See *program-name* LONG modifier, above).

Command Examples:

```
DUI> list

DUI> list short

DUI> list type utility

DUI> list xdiag long

DUI> list product HP1234

DUI> list product HP1234 short

DUI> list long
```

Output:

```
{For default or SHORT modifier:}

program name   program name   program name   program name
     :              :              :              :
     :              :              :              :
     :              :              :              :


{For complete program name:}

          Program   Program  Catalog                    Associated
  Name    Version   Type     Languages    Devices       Files
========  ========  =======  ===========  ============  ==========
cccccccc  c.cc.cc   cccc     cccccccccc   ccccccc ad*   cccccccc
                                          ccccccc       cccccccc
                                  :            :            :
                                  :            :            :
                                  :            :            :

* a  program is the auto-diagnostic for this device
  d  program is the hardware status decoder for this device
```

LISTREDO

The `LISTREDO` command displays the user's command history stack. Each command in the display will be numbered - these numbers may be used with the `DO` and `REDO` commands. If a number *n* is given as an argument to the command, only the *n* most recent commands in the history stack will be displayed. Otherwise, the entire history stack will be displayed.

Syntax:

```
LISTREDO [n]
```

Abbreviations and Alternate Tokens:

```
LR
```

Modifiers:

```
n  The number of commands in the history stack to
   be displayed. The n most recent commands
   will appear. If n is greater than the current
   size of the user's history stack the entire
   history stack will be displayed
```

Command Examples:

```
DUI> listredo

DUI> lr 3
```

Related Commands:

```
DO
REDO
```

MODE

The `MODE` command, used without a modifier, will tell the user the current mode of the operating system; either *single user* or *multi-user*. If one of the arguments is used, the operating system will be placed in the mode specified by the argument - assuming the user has the security capability necessary to place the system into that mode.

Syntax:

```
MODE [[=] SINGLE]
     [[=] MULTI ]
```

Abbreviations and Alternate Tokens:

none

Modifiers:

omitted	The current operating system mode will be displayed.
SINGLE	Place the operating system into single user mode. If the system is already in single user mode this modifier will have no effect. When an operating system is placed in single-user mode ALL other users are logged off.
MULTI	Place the operating system into multi-user mode. If the operating system is already in multi-user mode this modifier will have no effect.

Command Examples:

```
DUI> mode multi

DUI> mode

DUI> mode single
```

Related Commands:

SHOWSTATE

Limitations:

- The operating system mode will only be changed if the command is given by a user with diagnostic system security level 0.
- Placing an operating system in single-user mode causes all other users to be kicked out of the system - this could have serious, adverse affects especially on the popularity of the kicker.

MODIFY

The MODIFY command allows a user to change information about a diagnostic program without needing to reinstall the diagnostic. The changes specified will be permanent, but the message catalog, which contains the original information, will not be updated. That is, the original installation information in the message catalog will not be overwritten.

The user will be prompted for the location (file path) of the diagnostic system to be modified. This allows experimental/working subsystems to be built and maintained for development and integration purposes. The default will always be the location of the diagnostic system issued with the operating system.

The user will then be presented with a menu of modification tasks. Any or all of the tasks may be selected in any order. Any task may be selected more than once to correct the previous correction (except, of course, that CANCEL will not undo a previous cancel). Corrections will not actually be made final until the user selects DONE, at which point a list of the program information (same format as the LIST command) will be displayed.

If the changes are successfully finalized, a message indicating so will be generated and the user will be returned to the DUI prompt. If the changes could not be finalized, messages explaining the exact nature of the problem will be displayed before the DUI prompt reappears. The CANCEL task cancels **all** pending corrections that have not already been finalized with the DONE task. If DONE is selected and no corrections are pending, the MODIFY command simply terminates without modifying the program information.

The menu is only displayed once. However, the user may recall the menu by typing REFRESH at the MODIFY TASK prompt.

Default values for MODIFY prompts correspond to the pre-existing values for the program being modified.

The CATALOG task allows the user to add to and delete from the list of message catalogs for the specified program. A list of the languages of all the message catalogs associated with the diagnostic program will be displayed. The user will then be prompted to delete from this list. Next the user will be prompted for the locations and names of any message catalog files to be added to the list. The user will also be prompted for the name of the language the message catalog is written in.

The language of the message catalog must be specified in the language that the DUI is currently printing messages in. For example, if the DUI is currently using English and a German language catalog is being installed, the language of the catalog would be specified as being "German." If the DUI is currently using German, the language of a German language catalog would be specified as "Deutsche."

Installation data in the catalogs is **not** examined (as it would be if the INSTALL command were used); therefore, the current information for the program may not correspond to that in the catalogs. All catalog files must be the formatted output of the Native Language Support catalog formatter, GENCAT.

The CODE task allows replacement of the code for the specified program. The user will be prompted for the location of the file containing the executable code for the program as well as for the new version number. Note that a code change strictly requires a version change. Likewise, to modify the version number, the user must supply a new program file.

The DEVICES task allows the user to add to, delete from, and correct the list of devices which the program diagnoses (if applicable). A list of currently diagnosed devices will be displayed. Then the user will be prompted to add to, delete from, and correct the list. In addition to supplying the device name, the user will be able to specify if the program is the designated auto-diagnostic and/or hardware status decoder for that device. If any other program already serves as the designated auto-diagnostic/status decoder for the device, the user will be prompted to confirm an override of the previous designation. Prompts for additional devices will continue to appear until a simple carriage return (<cr>) is entered to terminate the prompt. When this happens, the user will be prompted for the names of any devices to be removed from the list. The user will be warned if deleting a device from the list of devices diagnosed by the program would result in that device no longer having a designated auto-diagnostic and/or hardware status decoder in the system. As with new devices, the user will continue to be prompted for additional devices until <cr> is entered.

The TYPE task allows the user to change the function type of the program. Valid options include DIAGNOSTIC, EXERCISER, UTILITY, and VERIFIER. These options may be abbreviated to a minimum number of characters.

The FILES task allows the user to add and delete extraneous files needed by the program such as files of downloadable code. A list of the files associated with the program will be displayed. Then the user will be prompted to delete from and add to the list.

The CANCEL task effectively destroys all changes made since entering MODIFY command.

The DONE task exits the MODIFY command. If any changes are pending, they are finalized.

The SHOW task displays the values for the data items as modified by any pending modifications. This command will produce results identical to the LIST command for the program being modified.

Syntax:

MODIFY *program-name*

Abbreviations and Alternate Tokens:

MOD

Modifiers:

program-name The name of the diagnostic program for which information is to be changed.

Command Examples:

```
DUI> modify xdiag

DUI> mod ydiag
```

Data Prompts:

```
Specify diagnostic system in which program will be modified:
[<cr>= supported system] >

   Modification Options:
          CATALOG  - Add or delete program message catalogs
          CODE     - Replace program code
          DEVICES  - Change list of diagnosed devices
          TYPE     - Change type of program
          FILES    - Add or delete special files

          CANCEL   - Cancel all pending changes and return
                     to modify menu

          DONE     - Implement changes and return to DUI
                     prompt

          EXIT     - Cancel all pending changes and return
                     to DUI prompt

          LIST     - Display pending changes

   MODIFY TASK >


   {for CANCEL task: }

        OK to cancel pending corrections? (YES/NO) [NO]  >


   {for CATALOG task: }

         Catalogs installed for this program are:
              - American
              - English
              - French
              - Canadian-French
              - Kanada
              - Katakana
              - German
```

```
Enter languages of catalogs which should be deleted:
   SYNTAX: language-name (as listed above)

    LANGUAGE [<cr>= no more languages] >
    LANGUAGE [<cr>= no more languages] >
    LANGUAGE [<cr>= no more languages] >
        :          :
        :          :


Need file containing program MESSAGE CATALOG and
  the language the file is written in ...
FILE NAME [<cr>= no more files] >
 LANGUAGE [<cr>= cancel this file] >
FILE NAME [<cr>= no more files] >
 LANGUAGE [<cr>= cancel this file] >
    :
    :
```

```
{for CODE task: }

     Need file containing executable PROGRAM code....
     FILE NAME [<cr>= cancel] >
     PROGRAM VERSION (v.uu.ff) >


{for DEVICES task: }
     Current devices installed for this program:
          - device v  AUTO
          - device x
          - device y  AUTO  DECODE
          - device z        DECODE

     Enter devices NO LONGER SERVICED by this program:
                            SYNTAX : device-name
               DEVICE  [<cr>= no more devices] >
               DEVICE  [<cr>= no more devices] >
               DEVICE  [<cr>= no more devices] >
                  :          :
                  :          :

     Enter NEW devices now serviced by this program
        OR CORRECTIONS to current devices
                            SYNTAX : device-name [Auto] [Decode]
               DEVICE  [<cr>= no more devices] >
               DEVICE  [<cr>= no more devices] >
```

```
                DEVICE  [<cr>= no more devices] >
                  :          :
                  :          :



{for TYPE task: }

     Program Type Options:
               DIAGNOSTIC
               EXERCISER
               UTILITY
               VERIFIER

     Specify program TYPE [cccccccccc] >


{for FILES task: }

     Current files:
               fileA
               fileB
                 :
                 :
                 :

     Enter files to be deleted
        FILE NAME [<cr>= no more files] >
        FILE NAME [<cr>= no more files] >
        FILE NAME [<cr>= no more files] >
              :         :
              :         :

     Enter files to be added
        FILE NAME [<cr>= no more files] >
        FILE NAME [<cr>= no more files] >
        FILE NAME [<cr>= no more files] >
              :         :
              :         :
```

Output:

```
    {for DONE task: }

The following MODIFICATION is about to be made to the
On-line Diagnostic Subsystem:


          Program   Program  Catalog                    Associated
  Name    Version   Type     Languages    Devices       Files
========  ========  =======  ===========  ============  ==========
cccccccc   c.cc.cc   cccc     cccccccccc   ccccccc ad*   cccccccc
                                          ccccccc       cccccccc
                                  :            :            :
                                  :            :            :
                                  :            :            :

* a  program is the auto-diagnostic for this device
  d  program is the hardware status decoder for this device


    {for SHOW task: }


          Program   Program  Catalog                    Associated
  Name    Version   Type     Languages    Devices       Files
========  ========  =======  ===========  ============  ==========
cccccccc   c.cc.cc   cccc     cccccccccc   ccccccc ad*   cccccccc
                                          ccccccc       cccccccc
                                  :            :            :
                                  :            :            :
                                  :            :            :

* a  program is the auto-diagnostic for this device
  d  program is the hardware status decoder for this device
```

Related Commands:

```
INSTALL
LIST
PURGE
```

Limitations:

- The installation information in the existing message catalog will not be updated.
- The installation information contained in a new message catalog will be ignored.

OUTFILE

The OUTFILE command causes all diagnostic system input and output to be written into the specified file. This is in addition to having the I/O appear on the user's terminal.

Outfiles are opened and processed so that each "line" in the file will consist of 80 characters. That is, each "line" has some combination of 80 characters (including spaces), followed by a carriage return, or by a line feed, or by a carriage return and a line feed (the line termination character(s) is operating system dependent). This file organization would be thought of as "80 byte fixed length record ASCII" on MPE/iX.

A user needn't do anything to cause an *outfile* to be organized in this manner. Conversely, a user cannot do anything to force the diagnostic system to write to *outfiles* in any other way.

Syntax:

```
OUTFILE [=] {filename}
            {OFF     }
```

Abbreviations and Alternate Tokens:

```
OF
```

Modifiers:

filename	The name of the file into which all I/O should be placed.
OFF	Stop copying I/O into the file.

Command Examples:

```
DUI> outfile yfoo

DUI> of xfoo

DUI> of off
```

Limitations:

- I/O will not be copied into a file if the name given is that of a pre-existing file.

PURGE

The PURGE command causes the named program to be removed from the diagnostic system. The catalog and program files for the named program are deleted from the location of the selected diagnostic system and all references to the named program are removed from diagnostic system files. All special files (such as downloadable code files) associated with the program will also be deleted.

The user will be prompted for the location of the diagnostic system from which the program is to be purged. Information about the program to be purged will be displayed (LIST command format) and the user will be prompted for a verification of the purge. If the diagnostic cannot be purged, a detailed explanation will be provided before returning to the DUI prompt. Otherwise, the user will be immediately returned to the DUI prompt.

Syntax:

PURGE *program name*

Abbreviations and Alternate Tokens:

none

Modifiers:

program name The name of the diagnostic program which is to be removed from the specified diagnostic system.

Data Prompts:

```
Specify diagnostic system from which program will be purged:
FILE PATH [<cr>= cccccccccccccc] >

{Where cccccccccccccc will be the file path to the
 group or directory where the "normal" or "default" diagnostic
 system resides}


The following program, along with all of its associated files and
message catalogs, will be removed from the diagnostic system:

          Program  Program  Catalog                        Associated
  Name    Version  Type     Languages     Devices          Files
========  ======== =======  ===========   ============     ==========
cccccccc   c.cc.cc  cccc    cccccccccc     ccccccc ad*      cccccccc
                                           ccccccc          cccccccc
                                 :             :               :
                                 :             :               :
                                 :             :               :

* a  program is the auto-diagnostic for this device
  d  program is the hardware status decoder for this device
```

```
Do you still wish to remove this program? [yes/no]
```

Command Examples:

```
DUI> purge xdiag
```

Related Commands:

```
INSTALL
MODIFY
```

REDO

REDO allows the user to edit and re-execute any command in the command line history stack. REDO is interactive. The DUI will display the command so that the user may edit it. After the user edits the command the DUI will display the modified command. This version of the command may also be edited by the user. This will continue until the user enters a carriage return without having done any more editing on the command line displayed by the DUI - the command will then be executed.

The first edit of the command may be specified with *edit-directives* attached to the REDO command. Subsequent edits of the command are done by typing editing directives under the displayed command (see examples).

Note

The arrow keys will not position the cursor when editing using the REDO command. The space bar and backspace keys must be used. Cursor position control characters (such as Control-H on HP-UX) may also be used, if the system recognizes them.

Syntax:

REDO [[CMD=] *command-id*] [, *edit-directives*]
[;EDIT= *edit-directives*]

Abbreviations and Alternate Tokens:

none

Modifiers:

command-id

Specifies the command to re-execute. The command may be specified by its relative or absolute order in the command line history stack, or by name (as a string) in whole or in part. The default *command-id* is -1, the most recent command. An error is detected if the *command-id* does not exist in the command line history stack.

The command represented by *command-id* will be displayed to the user (in a modified form if the command was given with *edit-directives*). The user may then modify the command again.

Each time the user modifies the command it will be re-displayed in its modified form. Command execution will not occur until the user hits a carriage-return without first having re-edited the command.

COMMAND-ID	EXECUTES
(omitted)	Previous command (same as REDO -1)
-n	The nth command before the most recent one. N is a number in the command line stack relative to the most recent command. The most recent command is -1.
m	Command number m in the command line stack. The number m is absolute.
string	The most recent command beginning with this string.

edit-directives

A string specifying changes to be made in the command represented by the *command-id* before it is displayed to the user. When the (edited) command line is displayed, the user may hit carriage-return to execute the command or may edit the command further by using the editing directives. Editing directives are placed under that part of the command string where the user wishes them to take effect.

If no *edit-directives* are given, the command represented by the *command-id* is displayed so the user may edit it as described above.

The *edit-directives* must be surrounded by quotation marks (" ") if they contain any scanner/parser delimiters such as: , ; " ' [ ] = or a space.

The editing directives which may be used as *edit-directives* are:

DIRECTIVE	EFFECT
i	INSERT. If text follows the i, the text following i is inserted in the current line at the position after the i.
r	REPLACE. If text follows the r, the text following r replaces the same number of characters in the

current line, beginning at the position of r.

d

DELETE. Deletes a character from the current line for each d specified in the edit line. Note that "d d" does not specify a range but simply deletes one character from the position above each d. Multiple d's may be followed by an insert or replace operation.

d>

DELETE. Deletes to the end of the current line from the position specified by d>. It may be followed by an INSERT or REPLACE operation.

>

APPEND. > followed by text appends the text to the end of the current line. If > is positioned beyond the end of the current line, then a replacement is performed instead.

>d

DELETE. Deletes from the end of the current line, right-to-left. Multiple d's may be specified after >, as well as INSERT and REPLACE strings.

>r

REPLACE. Replaces characters at the end of the command line. The replacement is done so that the last (rightmost) character of the replacement string is at the end of the line.

c

CHANGE. Changes all occurrences of one string to another in the current line when the search string and replace string are properly delimited. A proper delimiter is a non-alphabetic character: ', ", /, etc. The substitution is specified as:

c<delim> search-string <delim> [replace-string [<delim>]]

Omitting the replace-string causes occurrences of search-string to be deleted, with no substitution.

u

UNDO. A single u in column one cancels the most recent edit of the current line. Using the UNDO command twice in a row cancels all edits for the current line and re-establishes the original, unedited line. If u is placed anywhere other than column one of the current line, then a simple replacement is performed. UNDO makes sense only if you have a line on which you have performed some editing that can be "undone."

other	Simple replacement. Any other character (not i, r, d, d>, >, >d, c, or u) causes that character to be replaced in the current line at t position indicated by the character. In fact, simple replacement also occurs for the editing characters i, r, c, or > if they are not followed by text; or if > appears at or beyond the current end of line.

Command Examples:

EDITING SAMPLES

Practical uses of the editing commands listed above are shown here:

EDIT	ACTION
u	First occurrence undoes the previous edits. The u must be in column one.
u	Second occurrence undoes all edits on the current line. The u must be in column one.
rxyz	Replaces the current text with xyz starting at the position of r.
xyz	Replaces the current text with xyz starting at the position of x.
ixyz	Inserts xyz into the current line, starting at the position immediately before the i.
ddd	Deletes three characters, one above each d.
'd xyz'	Deletes a single character above the d, skips one space, then replaces the current text with xyz starting at the position of x.
ddixyz	Deletes two characters, then inserts xyz in the current line in the position before the i.
"d d"	Deletes one character above the first d, skips two spaces and deletes a second character above the second d. It does not delete a range of characters.
'd d>xyz'	Deletes a single character above the first d, skips two spaces and deletes to the end of the line beginning at the second d, and then appends xyz to the end of line.

>xyz	Appends xyz to the end of the current line.
>ddxyz	Deletes the last two characters from the end of the current line and then appends xyz to the end of the line.
>rxyz	Replaces the last three characters in the current line with xyz.
>ixyz	Appends xyz to the end of the line. In this case, the i command is superfluous, because > accomplishes the same result. Using >xyz would be sufficient.
c/ab/def	Changes all occurrences of ab to def, starting at c.
c"ab"	Deletes all occurrences of "ab" starting at c.
cxyz	Replaces the current text with cxyz, starting at c. Because no delimiters have been specified (as they were in the previous two examples), this is a simple replacement.

EXAMPLES

REDO pas	Edits the the most recent command beginning with the string pas.
REDO 10	Edits command number 10 (absolute) on the command history stack.
REDO -2	Edits the second-to-last command on the stack (one command before the most recent).
REDO , "c/$null/$STDLIST"	Change all occurrences of $null to $STDLIST in the most recent command before editing it.
REDO run, ">;debug"	Append ;debug to the the most recent :RUN command and then edit it.

Related Commands:

```
DO
LISTREDO
REDOLOAD
REDOSAVE
REDOSIZE
```

Limitations:

- REDO is based on an MPE/iX command.

REDOLOAD

REDOLOAD replaces the user's command history stack with the history stack which was saved by the command REDOSAVE.

Syntax:

REDOLOAD [*filename*]

Abbreviations and Alternate Tokens:

RDLD

Modifiers:

filename	The name of the history stack which was saved using the REDOSAVE command. If no file is specified, the default saved history file, **cmdhist**, will be restored.

Command Examples:

```
DUI> redoload

DUI> rdld histfile
```

Related Commands:

DO
REDO
REDOSAVE
REDOSIZE

Limitations:

- Attempting to restore a file which does not contain a history stack will cause unpredictable results. **cmdhist**, the default saved history file, will always contain the stack most recently saved - this might have been saved by another user.

REDOSAVE

REDOSAVE causes all or part of the user's command history stack to be saved into a file. This file may later be restored as the current history stack by using the **REDOLOAD** command. The command history stack file created by this command is a permanent file and so will continue to exist between operating system sessions.

Syntax:

REDOSAVE [FILE [=] *filename*] [*n*]

Abbreviations and Alternate Tokens:

RDSV

Modifiers:

FILE *filename*	The name of the file in which to save the command history stack. If specified, this must be a new file. If not specified, the command history stack will be saved into the default file, **cmdhist**, whether or not this file already contains a command history stack.
n	The number of commands in the current history stack to be saved by being written to a file. The *n* most recent commands will be saved. If *n* is greater than the number of commands actually in the current history stack or if *n* is not specified the entire history stack will be saved.

Command Examples:

```
DUI> redosave

DUI> rdsv 3

DUI> rdsv file foohist

DUI> rdsv 18 file fiehist
```

Related Commands:

DO
REDO
REDOLOAD
REDOSIZE

Limitations:

- If a file is named, it cannot be a pre-existing file. If no file is specified, the history stack will be saved into the default file **cmdhist**. overwriting the contents of this file.

REDOSIZE

REDOSIZE allows the user to set the maximum number of commands which will be saved in the user's command history stack. If this command is not used, the history stack will contain a maximum of 25 commands.

Syntax:

REDOSIZE [=] *n*

Abbreviations and Alternate Tokens:

RDSZ

Modifiers:

n The number of commands to be held in the user's command history stack. The *n* most recent commands will be retained.

Command Examples:

```
DUI> rdsz 15

DUI> redosize 12
```

Related Commands:

DO
REDO
REDOLOAD
REDOSAVE

Limitations:

- Setting the command history stack to an arbitrarily large size using this command could cause unpredictable results on some machines. Although no limit is placed on the number of commands which the history stack may hold, it is assumed that the user will limit the size to some "reasonable" number. The maximum "reasonable" size is operating system and machine dependent.

REPLY

REPLY is used to send a reply to the prompt of a diagnostic program which is running in the background.

If there is only one program running in the background, the *program-id* (a unique process identifier) need not be specified. If more than one program is running in the background and no *program-id* is specified, a list of the background programs waiting for replies will be given. Both the program-id and the programs prompt of every background program waiting for a reply will appear. The user may then repeat the REPLY command.

Not giving a *reply message* has the effect of sending a null string to the program as the reply.

Syntax:

REPLY [*program-id*] "*reply message*"

Abbreviations and Alternate Tokens:

REP

Modifiers:

program-id	The unique identifying number assigned to a process by the operating system.
reply message	The message to be sent to the background diagnostic. Could be a carriage return.

Command Examples:

```
DUI> reply yes

DUI> rep 12 no
```

Related Commands:

BACKGROUND

Limitations:

- The DUI does not check the reply message for validity - the message is sent to the diagnostic program exactly as it is typed.
- If a valid *program-id* is given without a *reply message*, the effect is to send a null string as the reply. A null string is also sent to the program if no *program-id* is given but only one program with a reply pending is running in the background.

RESUME

RESUME causes the execution of a previously suspended diagnostic program to continue. If the program which is to be resumed is not specified and the user has only one suspended program that one program will be resumed. A list of programs will be printed if more than one program is suspended and the user does not specify a *program-id*. The user should then repeat the command giving one or more of the *program identifiers* from this list.

Syntax:

```
RESUME [program identifier [[,]...]] [BACKGROUND]
       [ALL                        ] [BACKGROUND]
```

Abbreviations and Alternate Tokens:

RES

Modifiers:

program identifier	The diagnostic's identifier - the unique number assigned to the diagnostic program by the operating system.
ALL	Resume running all suspended programs.
BACKGROUND	Place the specified programs in the background - if they are not there already - and then resume them.

Command Examples:

```
DUI> resume

DUI> res

DUI> res 24 26 background

DUI> res all background
```

Limitations:

- The user may only resume programs which he "owns."
- "Resumed" programs which had been running in the foreground before being suspended will run in the foreground after being resumed. "Resumed" programs which were running in the background when suspended will run in the background when resumed.
- If *i*) more than one *program identifier* is given as an argument to the RESUME command or *ii*) the ALL argument is given but the BACKGROUND argument is not given and *iii*) more than one of the suspended programs to be resumed was running in the foreground when suspended then only the designated foreground process most recently suspended will resume. All other suspended foreground processes will remain suspended.

  All designated background suspended processes will be resumed.

RUN

The RUN command is used to start one or more diagnostic programs. It may be used explicitly or implicitly.

When more than one diagnostic program is named within a **run** command, all internal setup for each one of the programs will be performed before the DUI calls upon the operating system to launch the programs one after the other. The DUI will not wait for one program to complete before launching the next.

Syntax:

[RUN] {*program name* [*command modifier* [...]]} [,...]

Abbreviations and Alternate Tokens:

none

Modifiers:

program name The diagnostic program which is to be run.

command modifier The possible modifiers are

BACKGROUND
DEBUG
ERRCOUNT
ERRONLY
ERRPAUSE
ERRPRINT
HARDCOPY
INFILE
LDEV
LOOP
OUTFILE
PDEV
SECTIONS
TRACE
Diagnostic Specific Parameters

Please see the following pages for descriptions of each of these modifiers.

Command Examples:

```
DUI> run xdiag

DUI> xdiag

DUI> run xdiag &, ydiag pdev=4.0 sc 3, zdiag ldev 0 bg
```

Limitations:

- The number of diagnostic programs which can be launched with one *run* command is limited by the number of processes the operating system will allow a user.
- Only one diagnostic program may run in the foreground at any one time; when running multiple programs all but one must be run in the background.

RUN COMMAND MODIFIERS

The following pages present detailed functional specifications for the RUN command modifiers.

BACKGROUND

The BACKGROUND modifier causes the diagnostic program just invoked to be run in the background. Output from a program running in the background will appear on the user's terminal unless it is redirected by the use of the OUTFILE modifier.

Syntax:

```
BACKGROUND
```

Abbreviations and Alternate Tokens:

```
BG
```

Default Value:

```
off
```

Examples:

```
DUI> run xdiag &, ydiag bg
DUI> xdiag background outfile xout
```

Limitations:

- If the output of the program placed in the background is not redirected, it will appear on the user's terminal. This could cause confusion if more than one program is running in the background or if a program is also running in the foreground.
- When the output of a program running in the background is not redirected, prompts from the program will appear on the user's screen. These prompts CANNOT be replied to directly since there is no interaction possible with a program running in the background. The user must use the *reply* command to send responses to prompts from programs running in the background.

DEBUG

The DEBUG modifier causes the program to run within a debugger. If the user does not specify which debugger to use, the diagnostic system will invoke a default debugger such as the *nmdebug* debugger on MPE/iX or the *xdb* debugger on HP-UX.

Arguments specific to the debugger will not be accepted. This is an exception to the *Support User Interface Standard*.

If the DEBUG modifier has been associated with a diagnostic program by the **set** command, the user may state "debug off" on the run command line in order to run the diagnostic outside of the debugger that one time.

Syntax:

```
DEBUG [[=] OFF      ]
      [[=] debugger ]
```

Abbreviations and Alternate Tokens:

```
DB
```

Default Value:

```
off
```

Examples:

```
DUI> xdiag pdev 2/4.3 debug
DUI> run ydiag ldev ydg/0 debug
DUI> run zdiag debug off
```

Limitations:

- The DEBUG modifier will, normally, be used only for purposes of diagnostic program development and maintenance. Anyone using this modifier must ensure that the diagnostic program being invoked to run under the debugger has been compiled and linked with whatever options are necessary to get the executable file to run under the chosen debugger. Further, the user must see that any additional files needed to run under the debugger are on the system and in the appropriate places. For example, if the *xdb* debugger is used the source and header files for the program must be on the system.

ERRCOUNT

`ERRCOUNT` sets the number of errors to tolerate before aborting the diagnostic program.

Syntax:

`ERRCOUNT` *n*

Abbreviations and Alternate Tokens:

`EC`

Default Value:

infinite - an unlimited number of errors may occur

Examples:

```
DUI> xdiag ldev 4 errcount 12
DUI> run ydiag ldev 2 ec 5
```

Limitations:

- The program will be aborted automatically when the specified number of errors have occurred - the user will not have a chance to continue running the diagnostic, even if the modifier ERRPAUSE has also been set.
- If the number of errors to tolerate before terminating is specified (by using the *n* argument), the number MUST be equal to or less than maxint.

ERRONLY

When ERRONLY is on, only error messages will be printed; most informational messages generated by a diagnostic program will not be displayed.

ERRONLY may be placed "on" by typing ERRONLY or ERRONLY ON.

When ERRONLY is off, all messages will be printed - informational and error.

Syntax:

```
ERRONLY [[=] ON ]
        [[=] OFF]
```

Abbreviations and Alternate Tokens:

```
EO
```

Default Value:

```
off
If ERRONLY is on, the long message form is the default
```

Examples:

```
DUI> run xdiag erronly
DUI> run xdiag erronly on
DUI> set ydiag erronly
DUI> run ydiag ldev 3 erronly off
```

ERRPAUSE

When ERRPAUSE is on and an error occurs, the user will be queried as to whether to continue executing the program. If the user responds yes the program will continue. If the user responds no the program will be aborted.

The modifier ERRPAUSE is equivalent to ERRPAUSE ON.

Syntax:

```
ERRPAUSE [[=] ON ]
         [[=] OFF]
```

Abbreviations and Alternate Tokens:

```
EPS
```

Default Value:

```
off
```

Examples:

```
DUI> run xdiag pdev 2/4.2 errpause
DUI> ydiag ldev 4 errpause off
```

Limitations:

- If ERRCOUNT has also been set and the maximum number of errors reached, the program will abort without querying the user.

ERRPRINT

ERRPRINT may be used to control the number of messages printed when an error occurs.

When the LONG modifier is in effect, all error messages associated with an error will be printed - there may be duplicate messages and some of the messages may be obscure. By default, all error messages will be printed when an error occurs - LONG is the online diagnostics subsystem default.

When the SHORT modifier is in effect, only the error message stating what the diagnostic was trying to do, or what the diagnostic believes the problem to be will be printed.

ERRPRINT will always be ON in the online diagnostics subsystem; using the OFF modifier will have no effect.

Syntax:

```
ERRPRINT [ON   ]
         [OFF  ]
         [LONG ]
         [SHORT]
```

Abbreviations and Alternate Tokens:

```
EPR
```

Default Value:

```
on, long
In the online diagnostics subsystem, ERRPRINT cannot be turned off.
```

Examples:

```
DUI> run xdiag errprint          has no effect
DUI> run xdiag errprint on       has no effect
DUI> set ydiag epr short
DUI> run ydiag ldev=3 epr off    has no effect
DUI> xdiag errprint long
```

HARDCOPY

The `HARDCOPY` modifier causes all of the diagnostic program's input messages and output messages to be printed on a hardcopy device such as a line printer or laser printer.

This modifier does not redirect I/O - it just causes a hardcopy of it to be created. All I/0 will also appear on the user's terminal unless a modifier such as `INFILE` or `OUTFILE` has also been used.

The user may specify which hardcopy device to use by **`ldev`** or **`pdev`** but, normally, the diagnostic system will recognize a particular hardcopy device to be used by default.

The user may also specify an "environment" to be used by the hardcopy device. The environment is information which is used by the hardcopy device to control printing in some way. What this information is and how it must be stated varies from hardcopy device to hardcopy device and from operating system to to operating system. This argument is provided as a convenience for users who wish to control their printing environment and who are knowledgeable about the "environments" used by the chosen hardcopy device.

Syntax:

```
HARDCOPY [LDEV [=] logical device name [ENV [=] environment]]
         [PDEV [=] physical path        [ENV [=] environment]]
         [OFF                                                ]
         [ON                                                 ]
```

Abbreviations and Alternate Tokens:

```
HC
```

Default Value:

```
off
```

Examples:

```
DUI> ydiag hardcopy
DUI> run xdiag hc ldev 7
DUI> run zdiag hc off
```

INFILE

The INFILE modifier causes all input data expected by a diagnostic program to be read from the specified *infile* rather than from the standard input device.

Infiles are opened and processed under the assumption that each "line" in the file will consist of 80 or less characters, followed by a termination character or set of characters. The termination character is operating system dependent. . That is, each "line" has some combination of 80 characters or less (including spaces), followed by a carriage return, or by a line feed, or by a carriage return and a line feed, depending on the operating system. This file organization would be thought of as "80 byte fixed length record ASCII" on MPE/iX.

An *infile* with any "line" longer than 80 characters will not be read or processed correctly.

The OFF argument may be used if an *infile* was bound to the diagnostic using the *set* command, but the user does not wish to get input to the program from the file during the current run.

The modifier *INFILE* differs from the command *USEFILE*. A *USEFILE* contains a series of commands (and, possibly, data) and controls the diagnostic session until the end of file is reached. An *INFILE* contains data which a particular diagnostic program would expect the user to give it during the course of its (interactive) execution.

Syntax:

```
INFILE [=] {filename}
           {OFF     }
```

Abbreviations and Alternate Tokens:

```
IN
```

Default Value:

data is received via the user's terminal rather than a file

Examples:

```
DUI> run wdiag ldev 3 infile winput


The infile winput might contain the following:

     yes
     3
     continue
     yes
     exit

Assuming this is reasonable data for wdiag to receive during
the course of its execution.
```

LDEV

The LDEV represents the logical name of a device to be tested or used by a diagnostic. This name differs among the various operating systems. For example, on MPE/iX systems the LDEV is a number while on HP-UX systems the LDEV is the name of a special device file.

Syntax:

LDEV [=] *logical device name*

Abbreviations and Alternate Tokens:

none

Default Value:

```
There is no LDEV default value. If a device is needed for a
diagnostic program to run, that device MUST be specified
using either its PDEV or its LDEV.
```

Examples:

```
DUI> run xdiag ldev 6
DUI> ydiag sc 5 ldev dsk/c0d0
```

LOOP

The LOOP modifier specifies the number of times the sections and steps are to be repeated before the diagnostic program terminates. If the modifier is given without a number the sections and steps will be repeated until an interrupt is given.

The OFF argument has the same effect as setting LOOP to 1.

If the ERRPAUSE or ERRCOUNT modifiers are also set. their effect will take precedence over LOOP.

Syntax:

```
LOOP [[=] OFF]
     [[=] n  ]
```

Abbreviations and Alternate Tokens:

```
none
```

Default Value:

```
If LOOP is not set only one iteration of the sections and steps
will be performed before the diagnostic program terminates.
```

Examples:

```
DUI> run xdiag ldev 5 sc 4/6 steps 32,46,120/125 loop 6
DUI> xdiag ldev 5 loop
DUI> ydiag pdev 8.4.3 sc 7 loop off
```

Limitations:

- If LOOP is given without an argument, the sections and steps will be repeated infinitely. The only way to get out of the infinite loop is to send an interrupt and abort the diagnostic program. The program will terminate abnormally and with unpredictable results.
- If the number of loops is specified (by using the *n* argument), the number MUST be equal to or less than maxint.

OUTFILE

The OUTFILE modifier causes all output from a diagnostic program to be placed into the named file rather than be displayed on the user's terminal.

Outfiles are opened and processed so that each "line" in the file will consist of 80 characters. That is, each "line" has some combination of 80 characters (including spaces), followed by a carriage return, or by a line feed, or by a carriage return and a line feed (the line termination character(s) is operating system dependent). This file organization would be thought of as "80 byte fixed length record ASCII" on MPE/iX.

A user needn't do anything to cause an *outfile* to be organized in this manner. Conversely, a user cannot do anything to force the diagnostic system to write to *outfiles* in any other way.

The OFF argument may be used if an *outfile* was bound to a diagnostic using the *set* command but the user does not wish to have output placed in the file during the current run.

Syntax:

```
OUTFILE [=] {filename}
            {OFF     }
```

Abbreviations and Alternate Tokens:

```
OUT
```

Default Value:

```
Redirection of output does not occur; all output is displayed
on the standard output device which is usually the user's
terminal.
```

Examples:

```
DUI> run xdiag outfile fooout
DUI> ydiag out yout
```

Limitations:

- The file named cannot already exist.

PDEV

A PDEV represents the physical path to a device and is composed of numbers corresponding to hardware slot numbers with various sorts of punctuation separating the numbers. The punctuation used may be machine dependent.

Syntax:

PDEV [=] *physical path*

Abbreviations and Alternate Tokens:

none

Default Value:

```
There is no PDEV default value. If a device is needed for a
diagnostic program to run, that device MUST be specified
using either its PDEV or its LDEV.
```

Examples:

```
DUI> run xdiag pdev 4/2.3 sc 4
DUI> run ydiag pdev 8.1
```

SECTIONS

A SECTION is a major operation or set of related operations within a diagnostic which can be explicitly invoked by a user. Some or all of a diagnostic's SECTIONS may be designated by the diagnostic developer to be default SECTIONS which are to be run if the user does not explicitly invoke one or more SECTIONS. A SECTION which performs more than one operation will have a subset of those operations, the steps, designated as the defaults to be run when the section is named but none of its steps are specified. When the SECTION is invoked without explicitly stating which steps are to be run the "default" steps will be run automatically.

SECTIONS and steps are numbered although the mnemonics listed below may be used instead of explicit numbers. When SECTIONS are invoked they are run in numerical order.

Steps are also run in numerical order. Steps may be explicitly named, using numbers or mnemonics, only if their associated SECTIONS are also named.

The security capability of the user determines whether or not an operation represented by a section or step is actually performed. For example, a user would need a high security capability before the diagnostic system would allow him to perform an operation which could cause the loss of user data on a device.

If a SECTION includes more than one separately callable operation (step) the needed capability is determined and checked for each individual operation. If a user tries to run a section or step for which he lacks the appropriate security capability a message will be output stating which security capability is needed and what security capability the user has.

Any sections and steps which the user invoked for which he does have the appropriate security capability will be run. Any which require a higher user security capability will not be.

Syntax:

```
SECTIONS { | [=] [+]n[[,]...]  | } [(|[+]n[[,]...]  |)]
                 [-]                 [-]
           | [=] [+]n/n[[,]...]|    |[+]n/n[[,]...]|
                 [-]                [-]
```

Please note that steps are designated using either parenthesis () or square brackets []. The square brackets are not shown in the above syntax diagram.

The "+" and "-" before section and step numbers are used when the user has already set up a list of sections and steps to run using the set *command and when the user wishes to run "default" sections and or steps with certain exceptions. The "+" and "-" refer to additions to and subtractions from a list of default sections and steps. Using the "+" will cause the immediately following sections or steps to be executed IN ADDITION TO the previously set sections or steps. Using the "-" will cause previously set sections and steps to be executed with the EXCEPTION of those whose numbers are preceded by the "-".*

In any case, any number or number range preceded by the "+" will be run - security permitting - while any number or number range preceded by the "-" will not be run. The "+" and "-" may also be used with the `sections` *modifier in the* `set` *command to change the default sections and steps to be run when a diagnostic is invoked without needing to re-specify the entire list of wanted sections and steps.*

Abbreviations and Alternate Tokens:

```
SC
```

Default Value:

```
The sections and steps which the diagnostic writer
named as the defaults for the invoked diagnostic
program. This differs for every diagnostic program.
```

Examples:

```
DUI> run xdiag pdev 2/4 sections 5/6,7 (-9,20/50)
```

Note: in this example, step 9 will not be executed but steps 20 through 50 will be executed.

```
DUI> ydiag ldev 3 sc 4,6,+9 [10,+12]
```

Mnemonics:

```
The following may be used instead of section and/or step numbers.
However, the section or step numbers which will be executed when
one of these mnemonics is used varies from diagnostic to
diagnostic. Not all diagnostics will have sections and
steps associated with these mnemonics.

Each diagnostic writer determines which sections and steps will
be run when one of these mnemonics is given in conjunction with
a particular diagnostic.

For a list of the sections and/or steps which will
be run when one of these is used type
```

help *program name* **sections**.

These may be substituted for section numbers and step numbers

```
all
auto
clear
default
errlog
fast
hwstatus
id
loopback
non-destructive
selftest
interactive
```

Mnemonic Descriptions:

Please note that the actual affect of any of these mnemonics may be altered if the capability of the user is too low to allow some operation to be performed. Although the descriptions speak about sections the mnemonics may also be used to substitute for steps.

ALL: Run all sections, default and non-default alike.

AUTO: Run all sections which are designated as autodiagnosable. That is, run all sections which might be run automatically by the operating system when it detects a possible hardware defect.

CLEAR: Run whatever section or sections "clear" or "reset" the device being diagnosed.

DEFAULT: Run the default sections. This mnemonic will be useful when non-default sections have been specified at a more global scope and the user wishes to only run defaults locally.

ERRLOG: Run whatever sections read and decode error logs.

FAST: Run the sections which the diagnostic developer has designated as "fast." That is, run those sections which will quickly test a large part of the device.

HWSTATUS: Run the sections which read and decode the hardware status of a device.

ID: Run the sections which "identify" the device.

LOOPBACK: Run whichever sections perform a loopback to the device(s). The type of loopback(s) performed will vary from diagnostic to diagnostic - the diagnostic developer determines which loopback(s) to perform when this mnemonic is given.

NON-DESTRUCTIVE: Run only non-destructive sections or steps. This will be useful to users possessing the highest security level who wish to avoid inadvertently running any potentially destructive sections.

SELFTEST: Run the sections which perform selftests on the device(s).

INTERACTIVE: Run the diagnostic's interactive sections.

Examples of Mnemonic Use:

Please note that a user is not expected to know what section or step numbers a mnemonic replaces - the user may name a section or step number and the mnemonic that replaces it on the same line. This is not an error.

If there is a conflict among statements on the same line the rightmost statement will take precedence. Nested statements are not allowed.

`ALL:`

```
sc all

sc all(all)

sc all(+all)
```

Please note that the "+" and "-" operators may be used with any mnemonic even if, as in this case, they have no effect.

`AUTO:`

```
sc auto

sc +auto
```

Run previously named sections AND autodiagnosable sections.

```
sc 5/7(auto)
```

Only run those steps of sections 5 through 7 which are autodiagnosable.

```
sc 8(-auto)
```

Do not run the autodiagnosable steps of section 8. Do run the default steps *of section 8 which are not autodiagnosable.*

CLEAR:

`sc clear`

`sc clear(4, 10/12)`

Run steps 4 and 10 through 12 of whatever sections clear the device. If none of the sections which clear the device have any steps 4, 10, 11, or 12 nothing will be run and a message will be output to the user stating this.

DEFAULT:

`sc 9/24(1/300)`

Run steps 1 through 300 of sections 9 through 24.

`sc 14(default)`

Only run the default steps of section 14. These will be either the diagnostic system default steps or the steps which the user has designated to be defaults for section 14 using the `set` *command. The same effect would be achieved by using* `sc 14`.

`sc 10(+default)`

Run the default steps of section 10. The "+" has no effect here but is not an error.

`sc default`

Run all default steps of all default sections. The same effect can be achieved by invoking the diagnostic on a run command line without mentioning sections.

ERRLOG:

```
sc errlog

sc auto,errlog
```

Run the default steps of both the autodiagnosable and errlog sections.

FAST:

```
sc fast

sc 2/3,fast,6,10(-auto)
```

Run the default steps of sections 2, 3, and 6 and the default steps of all "fast" sections. Run the default steps of section 10 EXCEPT for those default steps which are also autodiagnosable. If section 10 is a "fast" section still only run its non-autodiagnosable default steps.

HWSTATUS:

```
sc hwstatus

sc auto, hwstatus(2/6)
```

Run the default steps of all autodiagnosible sections AND steps 2 through 6 of any hardware status sections. If any hardware status sections are autodiagnosible their default steps will be run.

ID:

```
sc id

sc id(-auto)
```

Run the non-autodiagnosable steps only of the "identify" sections. If the "identify" sections do not have any autodiagnosable steps the directive is ignored. If the "identify" sections are composed completely of autodiagnosable steps or are themselves autodiagnosable *sections no operations will be performed.*

LOOPBACK:

```
sc loopback

sc 3(-auto,+default), loopback(4/3,+auto)
```

If section 3 is a loopback section its autodiagnosable steps will be run since the "+auto" directive appears to the right of the "-auto."

NON-DESTRUCTIVE:

```
sc id

sc id(non-destructive)
```

Run the non-destructive steps only of the "identify" sections. If the "identify" sections are not divided into steps only run the sections if they are non-destructive.

```
sc non-destructive,5/8,3(+auto)
```

Although sections and steps will be run in numerical order the diagnostic user interface will accept them in any order. When section 3 is run in this example any steps previously set for it at a global level will be run along with all autodiagnostic steps. If no special steps have previously been set for it, its default and autodiagnostic steps will be run.

SELFTEST:

```
sc selftest

sc selftest(non-destructive), fast
```

INTERACTIVE:

```
sc interactive

sc 4/6(-auto), interactive,fast
```

The user is not expected to know which numbers correspond to each mnemonic so any of sections 4, 5, or 6 could be "interactive" and/or "fast" sections. Duplication is acceptable. If any of sections 4, 5, or 6 is either "interactive" or "fast" its autodiagnosable steps will be run if they are also default steps since the "interactive" and "fast" directives appear to the right of "-auto."

TRACE

The TRACE modifier enables/disables software tracing messages. The entering and exiting of procedures and functions are noted by the display of messages such as

```
Entering fee
  Entering fie
    Entering foe
      Entering fum
      Exiting fum
    Exiting foe
  Exiting fie
Exiting fee
```

Selective tracing may be done by using arguments corresponding to various types of software modules: diagnostic programs (PROG), diagnostic library routines (LIB), device access routines (DAR), and I/O system modules such as device drivers (SYS). Tracing of error conditions will occur if the ERROR modifier is used. Any of these may be used in combination.

Please note that the SYS argument will not result in trace messages on most systems. The "hooks" needed to make this work are only being placed in the appropriate code on a few systems.

Syntax:

```
TRACE [=] { OFF   }
          { ALL   }
          { PROG  }
          { LIB   }
          { DAR   }
          { SYS   }
          { ERROR }
```

Abbreviations and Alternate Tokens:

TR

Default Value:

off

Examples:

```
DUI> xdiag ldev 4 sc auto trace prog
```

```
DUI> ydiag ldev 2 tr off
```

Trace was probably "bound" to ydiag using the set *command. This turns tracing off during this one run of* ydiag.

```
DUI> run wdiag tr prog dar
```

Limitations:

- TRACE can only be effective when the code modules being traced are properly instrumented. The TRACE modifier cannot display progress through code which has not had the proper calls inserted during development.

Diagnostic Specific Parameters

Parameters, modifiers, or other information unique to a particular diagnostic program may, in some cases, be specified within the **run** command. Such information is placed within double quotes ("").

The information within the double quotes is passed directly to the invoked program - no checking is done by the DUI to determine the correctness or validity of the information being passed. Many diagnostic programs prompt for any specific information they need. This mechanism for passing information to a diagnostic is provided as a convenience to the diagnostics but is not used by all of them.

The **set** command may also be used to bind program specific parameters to a particular diagnostic. Every time that diagnostic is invoked the "set" information will be passed directly to the diagnostic.

Please see the diagnostic's manual regarding what information a particular diagnostic expects to receive in this manner.

Syntax:

"*information*"

Abbreviations and Alternate Tokens:

none

Default Value:

null

Examples:

DUI> run xdiag pdev 4.3.2 sc 7 (2/6,10) "*some information*"
DUI> set ydiag "*some information*"

Limitations:

- No checking is done before the information is passed to the diagnostic; the information is passed exactly as given.
- The information must be eighty (80) characters or less in length.

SETVAR

The SETVAR command allows the user to explicitly set the values of environmental variables which control various features of the user interface. The SETVAR command may also be used to examine the current values of these variables. The only environmental variable currently associated with the SETVAR command within the DUI is TRACE.

If SETVAR is given without a variable being specified, a list of all DUI environmental variables along with their current values will be displayed.

The variables modified by the SETVAR command affect the Diagnostic User Interface itself - NOT the diagnostic programs run from it.

Syntax:

SETVAR [*variable* [=] *value* [[,]...]]

Abbreviations and Alternative Tokens:

none

Modifiers:

variable	*value*
TRACE	OFF
	ALL
	PROG
	LIB
	SYS
	ERROR

```
The TRACE variable, used with the PROG modifier,
results in the display of software tracing messages.
The entering and exiting of the DUI's procedures and functions
are noted by the display of messages such as

         Entering fee
           Entering fie
             Entering foe
               Entering fum
               Exiting fum
             Exiting foe
           Exiting fie
         Exiting fee

Tracing of error conditions will occur if the ERROR modifier
is used. The errors which will be reported are diagnostic system
internal errors. Some of these "errors" are expected conditions
which are encountered during normal processing. The error messages
which will be displayed as a result of TRACE ERROR are written
for factory personnel troubleshooting the diagnostic system itself
```

- they are not written
for the end user and may be meaningless to many users.

Errors which would normally be reported to a user will continue to be reported whether TRACE ERROR is set or not.

PROG and ERROR may be used in combination.

Tracing will continue until the SETVAR TRACE OFF command is given or the diagnostic session is ended.

Command Examples:

```
DUI> setvar

DUI> setvar trace error

DUI> setvar trace prog

DUI> setvar trace error prog

DUI> setvar trace off
```

Limitations:

- The ALL, LIB, and SYS values of the SETVAR variable TRACE have no effect within the online diagnostics subsystem.

SET

The SET command allows the user to explicitly reset the system default values for *modifiers* which may be given on a **run** command line. Once set, these values are used for every run of every diagnostic during a diagnostic session unless the modifier values for an individual diagnostic are changed. These *modifier* values become the global defaults for the diagnostic session.

The SET command also lets the user bind particular command modifier values to individual diagnostic programs. Modifier values associated with an individual diagnostic through the use of the SET command become the default values for that diagnostic until the end of the diagnostic session.

Individual modifier values may be changed, temporarily, by naming those modifiers and their new values on the **run** command line of a diagnostic. At the conclusion of that one diagnostic run the modifier values will revert to their defaults.

If the SET command is given with a *program name* but no modifiers, a list of the current values of all possible modifiers to that *program* will be displayed.

If the SET command is given with neither *modifiers* nor a *program name*, the current global default value of each *modifier* will be displayed.

Syntax:

SET {[*program name*] *command modifier* [[,]...]}

Modifiers:

program name — The diagnostic program whose modifier default values are to be set or displayed.

command modifier — The modifier whose value is to be set at the global level. Any of the run command modifiers may be set in this way except the SECURITY modifier. The following is a complete list of the command modifiers which may be used with this command.

BACKGROUND
DEBUG
ERRCOUNT
ERRONLY
ERRPAUSE
HARDCOPY
INFILE
LDEV
LOOP
OUTFILE
PDEV
SECTIONS
TRACE
Diagnostic Specific Parameters

Please see the descriptions of each of these modifiers for an indication of values which may legitimately be assigned.

Command Examples:

```
DUI> set xdiag loop 5 errpause

DUI> set loop 20 errcount 4

DUI> set loop

DUI> set xdiag

DUI> set
```

Related Commands:

SHOWPARMS

SHOWACTIVE

The SHOWACTIVE command causes a list of all current diagnostic processes to be displayed along with their process identifier numbers. Each displayed diagnostic process is also noted as being "running," "suspended," or "aborting" and either "foreground" or "background."

A "running" program is one that is executing normally.

A "suspended" program is one which was "suspended" by use of the SUSPEND command and is waiting for a RESUME command to return to a running state.

An "aborting" program is one which is in the transient state occurring between the receipt of an ABORT command and the actual termination of the program.

"Foreground" and "background" show where the process is. Although every program will be designated as one or the other, the designation is most important for suspended programs. The behavior of the RESUME command may be affected by where a program was running when it was suspended.

Syntax:

```
SHOWACTIVE
```

Abbreviations and Alternate Tokens:

```
SA
```

Command Examples:

```
DUI> showactive

DUI> sa
```

SHOWDEFAULT

SHOWDEFAULT will cause a list of command modifiers and their default values to be displayed. The values displayed will be those the modifiers held when the diagnostic system was invoked; i.e.. the initial diagnostic system default values for user settable modifiers will be displayed.

If SHOWDEFAULT is given with a list of command modifiers, the diagnostic system default values for those modifiers will be displayed. If SHOWDEFAULT is given without a list of modifiers or with the ALL argument. all user settable modifiers will be displayed along with their default values.

Syntax:

```
SHOWDEFAULT [ALL                              ]
            [command modifier [[,]...]]
```

Abbreviations and Alternate Tokens:

SD

Modifiers:

ALL	List all user settable modifiers and their default values.
command modifier	A user settable diagnostic modifier. The following is a complete list of these modifiers. Any valid run command modifier except SECURITY and the "Diagnostic Specific Parameters" modifier may be used with the SHOWDEFAULT command.

BACKGROUND
DEBUG
ERRCOUNT
ERRONLY
ERRPAUSE
HARDCOPY
INFILE
LDEV
LOOP
OUTFILE
PDEV
SECTIONS
TRACE

Command Examples:

```
DUI> showdefault

DUI> sd all

DUI> sd loop errpause errcount
```

Related Commands:

SET
SHOWPARMS

SHOWPARMS

SHOWPARMS will cause a list of user settable modifiers and their current values to be displayed.

The arguments the user gives to this command will determine which set of values for the modifiers will be displayed. If no argument is given the effect is the same as using the ALL argument.

If no argument is given or if the ALL argument is given the global default modifier values will be displayed along with the values the modifiers have for individual diagnostic programs (the values set with the SET command).

Syntax:

```
SHOWPARMS [GLOBAL                ]
          [LOCAL                 ]
          [ALL                   ]
          [program name [[,]... ]]
```

Abbreviations and Alternate Tokens:

SP

Modifiers:

GLOBAL — Display all modifiers with the values they hold globally. Each of these values will be either the diagnostic system default value for the modifier or the value for the modifier which has been explicitly named using the SET command.

LOCAL — Display only those modifiers and values explicitly set for individual diagnostic programs.

ALL — Display all modifiers with the values they hold globally. Also display those modifiers and values which have been reset for individual diagnostic programs using the SET command.

program name — Display all modifiers with the values they hold for the named program(s). These values may be the original diagnostic system default values, the global values set with the SET command, or values set explicitly for the named program using the SET command. In any case, the values displayed will be those which will be used when the named program is invoked unless the values are temporarily reset on the run command line.

Command Examples:

```
DUI> showparms

DUI> sp xdiag ydiag

DUI> sp global

DUI> showparms local

DUI> sp all
```

Related Commands:

```
SET
SHOWDEFAULT
```

SHOWSTATE

The SHOWSTATE command will cause the current system mode (single-user or multi-user) and the user's security capability to be displayed.

Syntax:

```
SHOWSTATE
```

Abbreviations and Alternate Tokens:

```
SS
```

Command Examples:

```
DUI> showstate

DUI> ss
```

Related Commands:

```
MODE
```

SUSPEND

The SUSPEND command causes the execution of a diagnostic program to stop. The program will remain "frozen" until a RESUME command is received at which time program execution will resume.

If the SUSPEND command is given without an argument and only one program is running. that one program will be suspended. If more than one program is running and the SUSPEND command is given without an argument, a list of programs and their program identifier numbers (*program-id*) will be displayed. The user may then repeat the SUSPEND command giving one or more of these *program-ids* as arguments.

Syntax:

```
SUSPEND [ALL                     ]
        [program id [[,]... ]]
```

Abbreviations and Alternate Tokens:

SUS

Modifiers:

ALL	Suspend every running program which was invoked by the user.
program id	The unique number identifying a particular run of a particular program. Used to tell the diagnostic system which program(s) to suspend.

Command Examples:

```
DUI> suspend

DUI> sus all

DUI> sus 23 17
```

Related Commands:

RESUME

UNLOCK

The UNLOCK command is used to explicitly unlock a device. This will release the device back to general access. UNLOCK may be used to release a device back to the system for general usage after the device has been fixed (or replaced) if it had previously been locked because it was defective.

If no LDEV or PDEV is given with the UNLOCK command a list of locked devices will be displayed. The devices listed will be malfunction locked.

Syntax:

UNLOCK [LDEV [=] *logical device name*]
       [PDEV [=] *physical path*]

Modifiers:

omitted	A list of malfunction locked devices will be displayed.
LDEV *logical device name*	The logical name of the device.
PDEV *physical path*	The physical path to the device.

Command Examples:

```
DUI> unlock pdev 4/2.3

DUI> unlock ldev 12
```

USEFILE

The USEFILE command causes input to the diagnostic system to be gotten from the specified file rather than from a user's terminal. Reading from the file begins immediately; the file controls the diagnostic session until end of file is reached or the *usefile* is prematurely closed because an interrupt was received.

Usefiles are opened and processed under the assumption that each "line" in the file will consist of 80 or less characters, followed by a termination character or set of characters. The termination character is operating system dependent. . That is, each "line" has some combination of 80 characters or less (including spaces), followed by a carriage return, or by a line feed, or by a carriage return and a line feed, depending on the operating system. This file organization would be thought of as "80 byte fixed length record ASCII" on MPE/iX.

An *usefile* with any "line" longer than 80 characters will not be read or processed correctly.

Usefiles may be nested; that is, a *usefile* may contain the USEFILE command. All open *usefiles* will be closed when an interrupt is received.

Syntax:

USEFILE [=] *filename*

Abbreviations and Alternate Tokens:

USE

Modifiers:

filename	The file from which the diagnostic system should get its input.

Command Examples:

```
DUI> usefile foocmds

DUI> use cmdfile
```

The usefile *foocmds* might contain the following:

```
run xdiag pdev 4.0.12 sc 3/10
run ydiag ldev 3 infile foo outfile fum
list type utility
mode single
run wdiag ldev 0 sc 5 outfile fee
```

Limitations:

- The USEFILE command can only be run in the foreground.
- It is assumed that the file contains commands and input sensible to the diagnostic system. If not, errors may occur.

Error Messages

The following are the error messages generated by the DUI.

305 `*** COULD NOT READ MESSAGE AT DUI's PORT (DUIERR 305)`

CAUSE: Input, such as a command, was given to the DUI at a user's terminal. The input was queued to the DUI's message port. The attempt to pull the message off of the port so that it could be processed failed.

ACTION: Try again. If the failure occurs a second time get out of the diagnostic system any way possible. It may be necessary to log into another terminal and abort the DUI process - the EXIT command to the DUI is unlikely to work in the present case. Submit an SR against the DUI; give as much information as possible about the circumstances surrounding the error (was the command the first given during the diagnostic session or had previous input been processed correctly, had any unusual messages been printed when the DUI was invoked, etc.).

---

308 `*** RECEIVED UNEXPECTED MESSAGE AT DUI'S PORT (DUIERR 308)`

CAUSE: Either a program sent an unrecognized request (something other than enable_intr_notify, disable_intr_notify, suspend_prog, or req_user_info) or the message type of a message pulled off of the DUI's port was not one the DUI could recognize.

ACTION: Use the command SETVAR TRACE ERROR. Repeat the command which resulted in the error the first time and note the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the DUI noting all of the errors. If the command used was RUN, it is possible that the error is in the diagnostic being run rather than in the DUI. However, if in doubt, submit the SR against the DUI rather than against the diagnostic.

---

309 `*** COULD NOT READ FROM SET# ! MSG# ! (DUIERR 309)`

CAUSE: The attempt to retrieve a message from a diagnostic's message catalog failed. The most likely reason is that the message was never put into the catalog.

ACTION: Submit an SR against the diagnostic being run.

---

311 `*** UNRECOGNIZED PROCESS - NOT IN LIST OF CURRENT PROCESSES (DUIERR 311)`

CAUSE The DUI was either processing an output_data request, a program request, or a user reply to a program but could not find the process identifier (pid) in its (the DUI's) table of known processes. The pid is needed to process the request because, without it, the DUI has no way of knowing which catalog to pull the message to be printed from. (The DUI uses the pid as an "index" into a table which contains information about the process - including the file descriptor of the open message catalog belonging to the running program).

It is also possible that the pid was in the DUI's table, but the corresponding message catalog file descriptor was 0.

The pid is also needed when a "program reply" is processed. The reply received was a handshake to an interrupt received message which the DUI sent the program. If the pid associated with the handshaking cannot be found in the DUI's table the DUI cannot reset the "interrupt sent" flag for the process. Until this flag is reset, no more interrupts will be sent. (A moot point, since without the pid the DUI cannot know there is a process needing interrupt notification).

Lastly, the DUI will print this message when it has received a program request to enable_intr_notify, disable_intr_notify, suspend_prog, or request_user_info and cannot find the pid associated with the request (in the ipc message) in its table. Without the pid the DUI cannot process the request since the information needed to process the request is in the table.

The most likely reason for any of these situations to occur is that a timing problem in the underlying operating system caused the request or reply to be received by the DUI long after the DUI was informed that the diagnostic terminated. When the DUI is informed that a diagnostic is to terminate it always searches for and processes any messages associated with the diagnostic before processing the termination.

ACTION If the problem is reproducible, submit an SR against the DUI giving as much information as possible about what seems to be happening when the error occurs. If the problem ever occurs it is likely to be transient and difficult to reproduce - the SR should explain in as much detail as possible the context in which it happens.

312 *** COULD NOT CREATE A PORT FOR THE DUI (DUIERR 312)

CAUSE Call to build_port failed. Build_port returned an error status. (i.e. some status other than successful or dipc_dup_pname). If dipc_dup_pname had been returned, the DUI would just try again to build a port using a different port name.

ACTION There is something seriously wrong with either the inter-process communication system or with the diagnostic monitor. If the user can kill the monitor and its associated processes and remove the diagnostic ports and then restart the monitor the problem might go away. However, in most cases this will not be possible. Since the normal tracing and printing mechanisms have not been set-up this early in the code very little additional information can be gotten. Please log an SR against the DUI giving as much information as possible about what is happening. Include the operating system build version, whether the system had been recently rebooted or not, whether the diagnostic system had been run successfully before the problem appeared, etc.

---

319 *** COULD NOT DO INITIALIZATION NEEDED FOR PRINTING (DUIERR 319)

CAUSE Two possible causes.

1) could not form the file path for the DUI's message catalog

2) a message catalog for the DUI written in the system default language could not be found and the DUI was unable to determine if it was all right with the user to just use English language messages. (get_user_input returned with an unsuccessful status of some sort).

ACTION Check to see if a message catalog for the DUI is on the system. The catalog will be named CDUIFXXX where XXX is three digits corresponding to native language localization language codes. The default catalog is CDUIF000. If the catalog cannot be found or is not in the correct directory, have the system administrator put the file on the system with the correct access permissions. If the catalog is on the system, file an SR against the DUI.

---

320 *** COULD NOT PRINT - PRINTING INSTRUCTIONS INCOMPLETE (DUIERR 320)

CAUSE Neither a "print literal" nor a "print message from set" was specified when print_dui_msg was called. The DUI is the only code which uses this procedure.

ACTION Log an SR against the DUI giving a description of what was being done when the error was printed and a list of any messages which immediately preceded this one.

---

321 ***** COMMAND RECOGNIZED BY PARSING ROUTINES BUT NOT BY PROCESSING ROUTINES (DUIERR 321)**

CAUSE The command the user gave was in the table of command names recognized by the DUI's parser. However, it was not in the list of commands the DUI recognizes as being able to process.

ACTION Log an SR against the DUI listing this message and the command which was not recognized.

---

322 ***** INVALID LDEV SPECIFIED (DUIERR 322)**

CAUSE The ldev the user gave with the run command could not be converted into a pdev (which is actually used by the diagnostic system). The call to obtain_pdev returned a status of dac_invalid_ldev.

ACTION Determine the correct ldev (sysmap may be able to help if it is on the system) and run the diagnostic again or give the pdev instead of the ldev with the run command.

---

323 ***** COULD NOT CONVERT THE SPECIFIED LDEV INTO A PDEV (DUIERR 323)**

CAUSE The ldev the user gave with the run command could not be converted into a pdev (which is actually used by the diagnostic system). The call to obtain_pdev returned a non-successful status (some status other than dac_invalid_ldev or successful).

ACTION Determine the correct ldev (sysmap may be able to help if it is on the system) and run the diagnostic again or give the pdev instead of the ldev with the run command.

---

324 ***** FAILED TO SEND IPC MESSAGE TO A PORT (DUIERR 324)**

CAUSE Send_to_port failed (returned status something other than successful). The DUI was trying to send either one of the three program initialization messages or a user_interrupt notification to a diagnostic but failed for some unknown reason.

ACTION Re-run the diagnostic. If the failure occurs again, use the SETVAR TRACE ERROR command. Repeat the RUN command noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the DUI noting all of the errors.

---

325 `*** COULD NOT LAUNCH THE DIAGNOSTIC (DUIERR 325)`

CAUSE The diagnostic system service launch_process failed to launch a diagnostic. The message printed just prior to this one should give some indication why (the message will be the one associated with the error status returned by launch_process).

ACTION Whatever action is associated with the message printed immediately before this one.

---

326 `*** UNRECOGNIZED IPC PROGRAM FUNCTION (DUIERR 326)`

CAUSE The DUI's procedure send_msg_to_program was asked to process a request other than prog_info1, prog_info2, prog_info3, or user_interrupt. Therefore, the DUI did not recognize the request.

ACTION Re-run the diagnostic. If the failure occurs again, use the SETVAR TRACE ERROR command. Repeat the RUN command noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the DUI noting all of the errors.

---

327 `*** CANNOT RUN THE DIAGNOSTIC (DUIERR 327)`

CAUSE The DUI tried to split the file path for the diagnostic - just in case the run command included the fully qualified name of the diagnostic (the file name along with the complete path to it in the file system). This needs to be done before the DUI checks the list of installed diagnostics to see if the name is in it. The call to split_file_path failed (non-successful status returned).

ACTION Use the SETVAR TRACE ERROR command. Repeat the RUN command noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the DUI noting all of the errors.

---

328 `*** CANNOT OPEN THE CATALOG (DUIERR 328)`

CAUSE The DUI could not open the diagnostic's message catalog. (The call to the service cat_open returned a non-successful status). The most likely cause is that there is no message catalog for the diagnostic written in the native language currently being used on the system.

ACTION Make sure a properly generated message catalog for the diagnostic, written in the same language as the current system default language, has been installed.

---

329 `*** FAILED TO GET USER INPUT (DUIERR 329)`

CAUSE A call to get_user_input made from print_dui_msg failed. The **DUI** needed user input because the printing function set it received in print_dui_msg included a reply_pending function.

ACTION Use the SETVAR TRACE ERROR command. Repeat the same sequence of commands which led to the error noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the **DUI** noting all of the errors.

---

338 `*** ! COULD NOT BE SUSPENDED (DUIERR 338)`

CAUSE The request to the operating system to suspend the program failed. The program probably terminated between the the time the **DUI** checked to see if the program existed and the time the operating system received the suspension request.

ACTION If the error can be repeated, submit an SR against the **DUI**. Use the SETVAR TRACE ERROR command. Repeat the RUN command followed by the SUSPEND command noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the **DUI** noting all of the errors.

---

339 `*** ! COULD NOT BE RESUMED (DUIERR 339)`

CAUSE The request to the operating system to resume the program failed. The program probably terminated between the the time the **DUI** checked to see if the program existed and the time the operating system received the resumption request.

ACTION If the error can be repeated, submit an SR against the **DUI**. Use the SETVAR TRACE ERROR command. Repeat the RUN command followed by the RESUME command as before noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the **DUI** noting all of the errors.

---

340 `*** COULD NOT SUSPEND ! - IT IS NOT RUNNING (DUIERR 340)`

CAUSE The request to suspend a specified program failed because the program was not in a "running" state. The program is either already suspended or is aborting or terminating.

ACTION Nothing to be done.

---

341 `*** COULD NOT RESUME ! - IT IS NOT SUSPENDED (DUIERR 341)`

CAUSE The request to resume a specified program failed because the program was not in a "suspended" state.

ACTION Nothing to be done.

---

343 `*** THE SPECIFIED PROCESS IDENTIFIER IS NOT RECOGNIZED (DUIERR 343)`

CAUSE The specified pid is not in the DUI's process table, so it cannot be suspended, resumed, or aborted.

ACTION Use the SHOWACTIVE command to get a list of the pids of processes which may be suspended, resumed, or aborted.

---

344 `*** THERE ARE NO RUNNING PROCESSES TO SUSPEND (DUIERR 344)`

CAUSE The user gave the suspend command with the ALL option but the DUI's process table does not have any "running" processes in it so nothing can be suspended.

ACTION Nothing to be done.

---

345 `*** THERE ARE NO SUSPENDED PROCESSES TO RESUME (DUIERR 345)`

CAUSE The user gave the resume command with the ALL option but the DUI's process table does not have any "suspended " processes in it so nothing can be resumed.

ACTION Nothing to be done.

---

346 `*** THERE IS MORE THAN ONE RUNNING PROCESS. PLEASE SPECIFY BY PROCESS IDENTIFIER WHICH OF THE FOLLOWING SHOULD BE SUSPENDED (DUIERR 346)`

CAUSE The user gave the suspend command without an option, but the DUI's process table has more than one "running" process in it so the user must specify which process is to be suspended.

ACTION Re-enter the SUSPEND command naming by pid the process to be suspended.

---

347 *** THERE IS MORE THAN ONE SUSPENDED PROCESS. PLEASE SPECIFY BY PROCESS IDENTIFIER WHICH OF THE FOLLOWING SHOULD BE RESUMED (DUIERR 347)

CAUSE The user gave the resume command without an option but the **DUI**'s process table has more than one "suspended " process in it so the user must specify which process is to be resumed.

ACTION Use the SHOWACTIVE command to get a list of suspended processes and their pids. Re-enter the RESUME command naming the pid of the process to be suspended.

---

348 *** AN INTERRUPT WAS RECEIVED FROM THE USER TERMINAL. (DUIERR 348)

CAUSE A user interrupt was given at the **DUI** prompt.

ACTION Nothing to do.

---

349 *** ! COULD NOT BE ABORTED (DUIERR 349)

CAUSE The request to the operating system to abort the program failed. The program may have terminated between the time the **DUI** checked to see if it was running and the time the OS received the abort request.

ACTION Probably nothing to do. However, if this is repeatable (and not just the result of some very odd timing circumstance) submit an SR against the **DUI**. Use the SETVAR TRACE ERROR command. Repeat the same series of commands up to and including the ABORT command noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the **DUI** noting all of the errors.

---

350 *** THERE IS MORE THAN ONE RUNNING OR SUSPENDED PROCESS. PLEASE SPECIFY BY PROCESS IDENTIFIER WHICH OF THE FOLLOWING SHOULD BE ABORTED (DUIERR 350)

CAUSE The user gave the abort command without an option but the **DUI**'s process table has more than one process in it so the user must specify which process(es) is(are) to be aborted.

ACTION Use the SHOWACTIVE command to get a list of running and suspended processes along with the associated pids. Re-enter the ABORT command naming one or more of the pids.

351	`*** THERE ARE NO PROCESSES (DUIERR 351)`
CAUSE	The DUI's process table is empty. The user wanted to abort or suspend a process.
ACTION	There is nothing to be done.

354	`*** RESERVE WORD LIST MESSAGE ! COULD NOT BE OBTAINED (DUIERR 354)`
CAUSE	Call to CAT_READ to obtain the message containing a list of reserved words failed. A more specific error message will precede this one.
ACTION	Do the action associated with the message printed immediately before this one.

355	`*** UNABLE TO OBTAIN RESERVED WORD FOR ITEM ! FROM MESSAGE ! (DUIERR 355)`
CAUSE	The call to GET_TEXT to obtain a reserved word from a reserved word message failed. The DUI's catalog probably has a bug in it.
ACTION	Submit an SR against the DUI.

356	`*** MESSAGE ! WAS NOT ACCEPTED BY SCANNER PACKAGE (DUIERR 356)`
CAUSE	The procedure reset_scan failed while trying to accept the message given. This can usually only happen when an empty buffer is passed to the reset_scan function.
ACTION	Submit an SR against the DUI.

359	`*** UNABLE TO READ DUI PROMPT FROM MESSAGE CATALOG (DUIERR 359)`
CAUSE	The procedure was not able to successfully read the DUI prompt string from the message catalog. The fault might be in the procedure trying to read from the catalog or in the catalog itself.
ACTION	Submit an SR against the DUI.

---

363 ***** UNABLE TO READ DUI VERSION STRING FROM MESSAGE CATALOG (DUIERR 363)**

CAUSE The procedure was not able to successfully read the **DUI** version string from the message catalog. The fault might be in the procedure trying to read from the catalog or in the catalog itself.

ACTION Submit an SR against the **DUI**.

---

364 ***** UNABLE TO READ DUI HEADER FROM MESSAGE CATALOG (DUIERR 364)**

CAUSE The procedure was not able to successfully read the **DUI** header string from the message catalog. The fault might be in the procedure trying to read from the catalog or in the catalog itself.

ACTION Submit an SR against the **DUI**.

---

365 ***** UNABLE TO READ MONITOR VERSION FROM MESSAGE CATALOG (DUIERR 365)**

CAUSE The procedure was not able to successfully read the diagnostic monitor's version display string from the message catalog. The fault might be in the procedure trying to read from the catalog or in the catalog itself.

ACTION Submit an SR against the **DUI**.

---

366 ***** UNABLE TO READ DUI WARNING MESSAGE FROM CATALOG (DUIERR 366)**

CAUSE The procedure was not able to successfully read the **DUI** warning string from the message catalog. The fault might be in the procedure trying to read from the catalog or in the catalog itself.

ACTION Submit an SR against the **DUI**.

---

370 ***** UNABLE TO RETRIEVE MESSAGE (!) FROM SET (!) OF THE DUI MESSAGE CATALOG. (DUIERR 370)**

CAUSE The service used to pull messages from the **DUI**'s catalog failed to do so. The most likely reason (but not the only possible one) is that the message is not in the catalog.

ACTION The specific reason the message could not be gotten from the **DUI**'s catalog will be printed immediately before this message is. The action taken will depend on that message.

---

371 ***** UNABLE TO OBTAIN HELP INFORMATION FOR LDEV (!).**

(DUIERR 371)

CAUSE The ldev named by the user could not be translated into a corresponding pdev. The diagnostic system uses pdevs internally rather than ldevs. No processing can be done on any device nor can information be gotten about any device designated by ldev unless the diagnostic system can find a corresponding pdev.

ACTION Check to make sure that the ldev is correct and configured into the system. If it is, try again using the corresponding pdev rather than the ldev.

---

372 ***** UNABLE TO DETERMINE THE IDENTITY OF PDEV (!). FURTHER**

HELP INFORMATION CANNOT BE OBTAINED. (DUIERR 372)

CAUSE Could not get a help message for the specified pdev.

ACTION The reason the help message could not be gotten will be printed immediately before this message. Appropriate action depends on what that previous message is.

---

373 ***** UNABLE TO OBTAIN HELP INFORMATION FOR PDEV (!).**

(DUIERR 373)

CAUSE The user asked for the "id" of a particular pdev but the diagnostic system failed to find the product number of the device represented by the pdev.

ACTION The actual reason for the failure will be printed immediately before this message. Appropriate action depends on what that message is.

---

374 ***** UNABLE TO RETRIEVE MESSAGE (!) FROM SET (!) OF THE**

CATALOG FOR (!). (DUIERR 374)

CAUSE The service used to pull messages from diagnostic message catalogs failed to do so. The most likely reason (but not the only possible one) is that the message is not in the catalog.

ACTION The specific reason the message could not be gotten from the diagnostic's catalog will be printed immediately before this message is. The action taken will depend on that message.

---

375 `*** UNABLE TO OPEN THE MESSAGE CATALOG FOR (!).`
`(DUIERR 375)`

CAUSE Was unable to open a diagnostic's message catalog for reading. The most likely reason for this is that the catalog file has been accidentally purged from the diagnostic system. It is also possible that the diagnostic catalog has not been translated into the language currently being used by the diagnostic system.

ACTION Use the MODIFY command to reinstall the appropriate catalog. If the problem is that the catalog does not exist for the native language currently being used, either change the language being used to one for which a catalog exists or use the MODIFY command to install an existing catalog. When asked which language the catalog is written in, lie and give the current system language. This could cause problems for latter users, but will solve the immediate problem.

---

376 `*** UNABLE TO LOCATE THE MESSAGE CATALOG FOR (!).`
`(DUIERR 376)`

CAUSE Failed to fully qualify the message catalog name. (That is, the attempt to add the group and account or the directory path of the diagnostic system to the file name failed - the actual reason for the failure is unknown).

ACTION Use the command SETVAR TRACE ERROR. Repeat the command which resulted in the error the first time and note the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR noting all of the errors.

---

377 `*** (!) IS NOT CURRENTLY INSTALLED IN THE DIAGNOSTIC`
`SYSTEM. (DUIERR 377)`

CAUSE The diagnostic was not found in the list of currently installed diagnostics on the present system.

ACTION If a typographical error was made, just redo the command correcting the spelling of the diagnostic name. Otherwise, install the diagnostic.

---

378 `*** YOU HAVE A DIAGNOSTIC SECURITY LEVEL OF !, BUT A`
`MINIMUM DIAGNOSTIC SECURITY LEVEL OF ! IS NEEDED`
`TO PERFORM THE REQUESTED FUNCTION (DUIERR 378)`

CAUSE The user does not have sufficient security to do requested function.

ACTION Log in as another user (one who has the necessary security level) or ask the system administrator to add the appropriate capability to your account.

---

379 `*** THE DIAGNOSTIC PROGRAM'S CATALOG DOES NOT CONTAIN INFORMATION ABOUT SECTIONS NEEDED TO PROCESS THIS REQUEST(DUIERR 379)`

CAUSE In the RUN command, the user named sections without explicitly naming steps for every section named. The diagnostic's catalog does not have the default step message (Set 2, msg 3). Or the user named sections for a diagnostic which does not have any - the catalog does not have a section message.

ACTION Use the HELP command to find out if the diagnostic does have sections. If it does, file an SR against the diagnostic; state that the diagnostic's message catalog is missing the required set 2, message 3. If the diagnostic does not have sections, run it again without naming any sections.

---

380 `*** THE TEST SYSTEM IS RUNNING SO NO TRACING OF THE DUI (OTHER THAN ERROR TRACING) CAN BE DONE AT THIS TIME (DUIERR 380)`

CAUSE The user requested some trace of the DUI, but the test system is already handling a codetest or a duitest and so cannot be called upon (through start_test_system) to also do a trace without generating an error.

ACTION Try again after the codetest or duitest is finished. The codetest or duitest is being run by a user under a different DUI on the same machine.

---

381 `*** INAPPROPRIATE I/O REQUEST (DUIERR 381)`

CAUSE The DUI was sent an info string as its input (implying that the DUI was invoked programmatically). Input from the user is needed, but no infile or usefile is open so input would have to come from the terminal; however, it is an error to try to get input from the terminal if an info string has been received - the DUI should just exit rather than getting more input at this point.

ACTION Create an infile or usefile with the input the DUI needs and correct the programmatic call to the DUI so that the command received in the info string refers to the infile or usefile.

---

501 `*** SYNTAX ERROR (DUISERR 501)`

CAUSE Something was syntactically wrong with a command or a sections/steps message in a diagnostic's catalog. The actual error may be printed out before this message is printed.

ACTION Re-enter the command correctly (use HELP <command> SYNTAX to see the correct syntax). If the RUN command was used and the problem appears to be with the sections/steps, submit an SR against the diagnostic stating that there is a syntactical error in one of the sections/steps messages. Give an exact copy of the command as typed in the SR so that the engineer can narrow down the the problem to the most likely catalog message.

505	`*** CANNOT PARSE MESSAGE CATALOG MESSAGE (DUISERR 505)`
CAUSE	A sections/steps message in a diagnostic's message catalog has an open parenthesis ("(") without a matching closing parenthesis (")").
ACTION	Submit an SR against the diagnostic explaining the problem. Give an exact copy of the command as originally typed in the SR so that the engineer can narrow down the problem to the most likely catalog message.

509	`*** INVALID RANGE - ENDING VALUE < THAN STARING VALUE (DUISERR 509)`
CAUSE	A section or step number range of the form n/n was given in the RUN command or found in one of the diagnostic message catalog section/step messages. The second number in the range was smaller than the first number.
ACTION	If the range was given in the RUN command, re-enter the command using correct numerical values in the range. If the RUN command was correct, submit an SR against the diagnostic explaining the problem. Give an exact copy of the command as originally typed in the SR so that the engineer can narrow down the problem to the most likely section/step catalog message.

513	`*** NO INPUT RECEIVED (DUISERR 513)`
CAUSE	A blank line rather than a command was given by the user. Not really an error.
ACTION	Ignore message and continue.

515	`*** LINKED LIST BEING CHECKED OR MANIPULATED IS EMPTY (DUISERR 513)`
CAUSE	One of the DUI procedures which adds to or removes a section or step from the linked list which the DUI uses internally to determine which sections and steps the user wishes to run was passed a null linked section/step list.
ACTION	Use the command SETVAR TRACE ERROR. Repeat the command which resulted in the error the first time and note the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the DUI noting all of the errors.

---

522 ***** NEEDED A CLOSING PARENTHESIS ")" (DUISERR 522)**

CAUSE A set of steps in a diagnostic's message catalog section/step message or a step given with the HELP STEP command was preceded by an open-parenthesis ("(") but had no matching closing parenthesis (")").

ACTION If the HELP STEP command was at fault, re-enter the command with a closing parenthesis. Otherwise, enter the SETVAR TRACE ERROR command. Repeat the command which resulted in the error the first time and note the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the diagnostic noting all of the errors.

---

525 ***** NEEDED A CLOSING SQUARE BRACKET "]" (DUISERR 525)**

CAUSE The RUN command was given with an open square bracket ("[") used to signify the beginning of a list of steps. The step list was not terminated with a matching closing square bracket ("]").

ACTION Re-enter the RUN command correctly.

---

530 ***** PRINTING ENVIRONMENT EXPECTED (DUISERR 530)**

CAUSE The "env" keyword was given with either the HARDCOPY command or the RUN command with a HARDCOPY parameter, but no printing environment was specified.

ACTION Redo the command either specifying a printing environment or without the "env" keyword.

---

533 ***** EXPECTED THE NAME OF A FILE (DUISERR 533)**

CAUSE A command which requires a file name or a diagnostic name was given without naming a file or diagnostic.

ACTION Redo the command, giving the appropriate file name or diagnostic name. Enter the HELP <command> or HELP <command> SYNTAX for more information.

---

536 ***** LDEV SPECIFICATION EXPECTED (DUISERR 536)**

CAUSE The "ldev" keyword was given with a command, but no ldev was named.

ACTION Redo the command specifying an appropriate ldev.

---

538 ***** PDEV SPECIFICATION EXPECTED (DUISERR 538)**

CAUSE The "pdev" keyword was given with a command, but no pdev was named.

ACTION Redo the command specifying an appropriate pdev.

541	`*** FILE NAME OR "OFF" EXPECTED HERE (DUISERR 541)`
CAUSE	The OUTFILE command was given or the RUN command was given with an OUTFILE parameter but without naming an outfile or specifying "off."
ACTION	Redo the command either specifying an outfile or "off."

545	`*** "OFF" OR A NUMBER EXPECTED HERE (DUISERR 545)`
CAUSE	The RUN command was given with the "loop" parameter, but neither a number nor the word "off" was specified.
ACTION	Re-enter the command specifying either a number or the option "off."

546	`*** "ON" OR "OFF" EXPECTED (DUISERR 546)`
CAUSE	The RUN command was given with either the "erronly" or "errpause" parameter, but the parameter was not followed by either the word "on" or "off."
ACTION	Re-enter the command specifying either "on" or "off" after the parameter.

553	`*** TRACING OPTION EXPECTED HERE (DUISERR 553)`
CAUSE	The "trace" parameter was given with the SETVAR or RUN command but no tracing option was specified.
ACTION	Re-enter the command specifying a tracing option. Use the HELP TRACE command to get a list of tracing options.

555	`*** QUOTED TEXT STRING EXPECTED HERE (DUISERR 555)`
CAUSE	Diagnostic specific parameters were assumed to be given with the RUN command since a double-quote (") was found in the command. However, there was something wrong with the quoted string - it might not have had a terminating quote or might have been null or might have had some other problem.
ACTION	Re-enter the command, correcting the string parameter.

560	`*** APPROPRIATE MNEMONIC MESSAGE COULD NOT BE FOUND IN THE DIAGNOSTIC'S CATALOG (DUISERR 560)`
CAUSE	The user gave mnemonics in place of some section(s) or step(s) in the RUN command, but no message could be found in the diagnostic's message catalog which would allow the DUI to translate the mnemonic into the corresponding numbers.
ACTION	Re-enter the RUN command using section/step numbers instead of mnemonics. A list of the sections and steps in the diagnostic may be gotten by using the HELP <diagnostic name> SECTIONS command.

563 `*** COULD NOT FIND THE NUMBER BEING SEARCHED FOR IN THE CATALOG MESSAGE (DUISERR 563)`

CAUSE The DUI tried to remove a section or step number from the linked list of sections/steps it maintains while determining which sections and steps the user wishes to run with a diagnostic. The section/step number was not in the list. This is not a real error condition.

This happens when the DUI's list has been made using default sections and/or steps. The number not found was not found because it was not in the diagnostic's catalog message containing default sections or steps.

The user should never see this message.

ACTION Ignore the message if the diagnostic runs after the message appears. If the diagnostic does not run, submit an SR against the DUI stating that this error message appeared and giving an exact copy of the RUN command which resulted in the message appearing.

---

565 `*** NEITHER A NUMBER NOR A MNEMONIC WAS FOUND (DUISERR 565)`

CAUSE A number or mnemonic was searched for but not found in either a diagnostic catalog section/step message or in a command given by the user. This is not necessarily an error; for instance, it might just signify that the end of a list of sections or steps had been reached. However, this message will only be printed when the condition is an error - as when the user gives the RUN command with the "sections" parameter but fails to name any sections.

ACTION Re-enter the command, specifying a number where needed.

---

570 `*** EXPECTED A NUMBER OR MNEMONIC AFTER THE COMMA (DUISERR 570)`

CAUSE A number list was given with a command for which such a list is appropriate. One of the numbers was followed by a comma but no number appeared after the comma. If the RUN command was given, either a number or a mnemonic would have been acceptable after the comma, but neither appeared.

ACTION Re-enter the command. Either specify a number after the comma or do not type the last comma.

---

573 `*** EXPECTED A NUMBER AFTER THE RANGE SIGN (DUISERR 573)`

CAUSE A command was given with a number range ("n/n"), but the last number in the range was missing. No number followed the range sign so what was read was of the form "n/".

ACTION Re-enter the command specifying the last number in the range.

---

574 *** EXPECTED A NUMBER OR MNEMONIC AFTER THE SIGN (DUISERR 574)

CAUSE A "+" or "-" sign was found while trying to parse a command (most likely the RUN command), but the sign was not followed by a number or mnemonic.

ACTION Re-enter the command either specifying a number or mnemonic after the sign or eliding the sign.

---

583 *** "OFF" OR A DEBUGGER NAME EXPECTED HERE (DUISERR 583)

CAUSE The RUN command was given with the "debug" parameter, followed by an equals sign ("debug="), but neither a debugger name nor the word "off" was found after the sign.

ACTION Re-enter the command either specifying a debugger or giving the "off" keyword. The "debug" parameter may also be given without the equals sign - signifying that the default debugger should be used.

---

585 *** "OFF," "ON," "LONG," OR "SHORT EXPECTED HERE (DUISERR 585)

CAUSE The RUN command was given with the "errprint" parameter, followed by an equals sign ("errprint ="). But, none of the possible options to "errprint" was found after the equals sign.

ACTION Use the HELP ERRPRINT command to see a list of all possible options and their use. Re-enter the command either naming an option after the "errprint" parameter or omitting the equals sign.

---

592 *** PROGRAM IS NOT INSTALLED (DUISERR 592)

CAUSE What was assumed to be a diagnostic name given with the RUN or HELP command was not found in the diagnostic system's list of installed diagnostics.

ACTION If a typographical error was made re-enter the command. If the diagnostic is not installed, install it. A list of installed diagnostics may be gotten by using the LIST command.

---

593 *** OPTION IS INVALID FOR THIS COMMAND (DUISERR 593)

CAUSE A command was given containing a keyword which is valid for another command, but not for the given command.

ACTION Re-enter the command using the correct syntax. The valid form for the command may be seen by using HELP <command> or HELP <command> SYNTAX.

---

595 *** THE SECTION NUMBER WAS NOT FOUND IN THE MESSAGE CATALOG MESSAGE (DUISERR 595)

CAUSE The DUI was unable to find a particular section number (probably given in the RUN command) in the diagnostic's message catalog message of valid sections.

ACTION Use the HELP <diagnostic> SECTIONS command to see a list of valid sections for the diagnostic. Re-enter the command.

---

598 *** UNEXPECTED TOKEN FOUND (DUISERR 598)

CAUSE While trying to parse a diagnostic catalog section/step message an unexpected punctuation mark was found (something other than a slash, open-parenthesis, close-parenthesis, or comma).

ACTION Use the SETVAR TRACE ERROR command. Repeat the RUN command noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the diagnostic noting all of the errors.

The RUN command may also be redone naming the sections in some alternate way. That is, if the original command used number ranges try listing the numbers explicitly. If the original command used mnemonics, try the numbers instead. If the original command relied on the defaults (i.e. didn't specify numbers or mnemonics), explicitly name the sections/steps to be run. The diagnostic catalog section/step messages which the DUI must parse in order to determine which sections/steps to run are dependent on the form of the input used in the RUN command. By changing the form of input, it is possible that the message with the error can be avoided. An SR should still be submitted though even if this works.

---

599 *** UNRECOGNIZED INPUT (DUISERR 599)

CAUSE A command was given which contained unrecognized garbage. Usually something extra and unexpected was found at the end of the command line. This could also appear if a valid parameter which can only be used once with the command is repeated.

ACTION Use the HELP <command> or HELP <command> SYNTAX to see the valid form of the command. Re-enter the command.

---

600 *** PROGRAM IDENTIFIERS ALREADY GIVEN - "ALL" NOT VALID (DUISERR 600)

CAUSE The ABORT command was given with the program identifiers of the programs to be aborted. Somewhere after the identifier list, the word "all" appeared. "All" is not valid if a program identifier has already been named.

ACTION Redo the command, eliding the "all."

---

601 *** "ALL" ALREADY GIVEN - PROGRAM IDENTIFIERS ARE NOT VALID (DUISERR 601)

CAUSE The ABORT command was given with the "all" parameter. Somewhere after the "all" a program identifier was named. Program identifiers are not valid if the "all" parameter has already been used.

ACTION Redo the command, deleting the program identifier(s).

---

602 *** A BLANK IS NOT VALID AFTER THE ASTERISK (DUISERR 602)

CAUSE A hardcopy environment was given beginning with an asterisk, but a space or tab appeared after the asterisk - this is not allowed.

ACTION Redo the command, removing the blank or tab.

---

603 *** A TYPE (DIAGNOSTIC, EXERCISER, VERIFIER, OR UTILITY) IS NEEDED (DUISERR 603)

CAUSE The user gave the LIST command with the "type" parameter but did not specify which type he wanted information about.

ACTION Use the HELP LIST or HELP LIST SYNTAX command to see a list of valid list types. Re-enter the LIST command.

---

604 *** A PRODUCT NAME IS NEEDED (DUISERR 604)

CAUSE The user gave the LIST command with the "product" parameter but did not specify which product he wanted information about.

ACTION Use the HELP LIST or HELP LIST SYNTAX command to see a list of valid products. Re-enter the LIST command.

---

605 *** A PROGRAM NAME WAS EXPECTED AFTER THE COMMA (DUISERR 605)

CAUSE The LIST command was given with program names. A comma followed one of the program names but was not itself followed by a program name in turn.

ACTION Redo the LIST command, either adding a diagnostic program name after the offending comma or omitting the comma.

---

606 *** A PARAMETER WAS EXPECTED AFTER THE EQUALS SIGN (DUISERR 606)

CAUSE An equals sign appearing after a command was not followed by a parameter.

ACTION Use HELP <command> or HELP <command> SYNTAX to see the valid form for the command. Re-enter the command.

---

607 ***** A COMMAND IDENTIFIER WAS EXPECTED AFTER THE EQUALS SIGN (DUISERR 607)**

CAUSE The DO or REDO command was given with the "cmd=" parameter. However, no command identifier appeared after "cmd="

ACTION Redo the command, specifying a command identifier. Use the HELP DO or HELP REDO commands for a complete explanation of valid input.

---

608 ***** AN EQUALS SIGN IS EXPECTED AFTER "CMD" AND "EDIT" (DUISERR 608)**

CAUSE The DO or REDO command was given with the "cmd=" and/or the "edit=" parameter. However, the "=" was missing. The equals sign is not optional in this case.

ACTION Redo the command, inserting the equals sign.

---

609 ***** EDITING DIRECTIVES WERE EXPECTED HERE (DUISERR 609)**

CAUSE The DO or REDO command was given without specifying editing directives after naming one of the parameters which must be followed by editing directives (";edit" or ",").

ACTION Re-enter the command, specifying editing directives. Use the HELP DO or HELP REDO command for information about editing directives.

---

610 ***** A STRING WAS EXPECTED HERE (DUISERR 610)**

CAUSE The DO or REDO command was given with the "cmd=" and/or ";edit" or "," parameter tokens. A legitimate command identifier/edit directive was not found after the parameter.

ACTION Use the HELP DO or HELP REDO command to see a discussion of command identifiers and editing directives. Redo the DO or REDO command.

---

611 ***** THIS STRING DOES NOT HAVE A TERMINATOR. (DUISERR 611)**

CAUSE The DO or REDO command was given with a quoted string as a command identifier or editing directive. However, no terminating single or double quote was found to match the quote with which the string began.

ACTION Redo the command, adding the terminating quotation character.

---

612 *** EXPECTED A NUMBER AFTER THE EQUALS SIGN.
(DUISERR 612)

CAUSE The HELP <diagnostic> SECTION or HELP <diagnostic> SC command was given with an equals sign immediately following the SECTION or SC. No number appeared after the equals sign.

ACTION Redo the command either inserting a number after the equals sign or deleting the equals sign.

---

613 *** EXPECTED A NUMBER HERE. (DUISERR 613)

CAUSE A number was not found in a command when one was required.

ACTION Use HELP <command> or HELP <command> SYNTAX for information about the correct form of the command. Redo the command.

---

614 *** NEED A PROGRAM NAME (DUISERR 614)

CAUSE The RUN command was given, but no program was named.

ACTION Redo the command, inserting the name of an installed diagnostic program. A list of installed programs may be gotten by using the LIST command. The proper form for the RUN command may be gotten by using the HELP RUN command.

---

616 *** THIS CANNOT BE A PROGRAM IDENTIFIER (DUISERR 616)

CAUSE A command was given with a program identifier as a parameter. But the program identifier given was too large to be a legitimate pid.

ACTION Use the SHOWACTIVE command to get a list of active diagnostic processes and their identifiers. Re-enter the original command giving the correct pid.

---

802 *** CAN'T READ THE TEMPORARY DIAGNOSTIC LIST FILE.
(DUIIERR 802)

CAUSE This error means that the temporary diagnostic list file could not be read from. The system may be corrupted at this point.

ACTION Submit an SR explaining the problem.

---

803 *** THE FILE PURGE LIST FILE COULD NOT BE OPENED.
(DUIIERR 803)

CAUSE This means the temporary purge file could not be opened. The file should exist.

ACTION Submit an SR explaining the problem.

---

805 `*** THE FILE PURGE LIST FILE COULD NOT BE CLOSED. (DUIIERR 805)`

CAUSE The error means that the temporary purge file could not be closed.

ACTION Submit an SR explaining the problem.

---

806 `*** CAN'T DELETE THE FILE !. (DUIIERR 806)`

CAUSE This means that a file that was involved with a deleted diagnostic could not be deleted. File protected may have been altered or the file actually doesn't exist which means the directory has been corrupted.

ACTION You can delete the file by hand if it still exists but there is no reason that the file should have a problem being deleted. If you can't, submit an SR explaining the problem.

---

807 `*** CAN'T DELETE THE TEMPORARY PURGE LIST FILE. (DUIIERR 807)`

CAUSE This means that the temporary purge file could not be deleted.

ACTION Submit an SR explaining the problem.

---

808 `*** THE FILE PURGE LIST FILE COULD NOT BE READ. (DUIIERR 808)`

CAUSE This means the temporary purge file was opened but can't be read from.

ACTION Submit an SR explaining the problem.

---

902 `*** THE TEMPORARY DIAGNOSTIC LIST FILE HAS BEEN CORRUPTED. (DUIIERR 902)`

CAUSE This means that a copy of the temporary diagnostic list file was found to be corrupted during a modify catalog or modify file command.

ACTION Submit an SR explaining the problem.

---

903 `*** THE VERSION ! IS INVALID. TRY AGAIN. (DUIIERR 903)`

CAUSE The program version in a catalog or one input from a modify code command has an invalid syntax. The syntax should be a.bb.cc. If the modify code command was chosen the version may be lower than the previous version.

ACTION A higher version should be entered if in a modify code command otherwise the version in the catalog should be corrected.

---

904 `*** CAN'T DELETE THE TEMPORARY MODIFY DIAGNOSTIC LIST FILE. (DUIIERR 904)`

CAUSE This means that a temporary diagnostic list file could not be deleted during a modify command.

ACTION Submit an SR explaining the problem.

---

908 `*** ! IS ALREADY TESTED BY !. (DUIIERR 908)`

CAUSE This means that a device which is already being tested by the program is being added to the list of devices the program tests. If this occurs during an install, then a duplicate device name exists in the device list.

ACTION Please re-enter a device that is not tested by this program.

---

911 `*** THE ! CATALOG ALREADY EXISTS. (DUIIERR 911)`

CAUSE This means that a catalog is being added to the list of catalogs belonging to the program but the program already has a catalog with that language. If this is during an install then the catalog in the DUI's language needs to be edited.

ACTION Please re-enter the language.

1001 `*** THE TEMPORARY DIAGNOSTIC DIRECTORY FILE HAS BEEN CORRUPTED. (DUIIERR 1001)`

CAUSE This means the copy of the diagnostic directory has been corrupted. It could also mean the real diagnostic directory is corrupted.

ACTION Submit an SR explaining the problem.

1201 `*** THE CATALOG ROOT COULD NOT BE FORMED (DUIIERR 1201)`

CAUSE This means the catalog root in the diagnostic directory could not be converted to a string.

ACTION Submit an SR explaining the problem.

1202 `*** THE CATALOG NAME COULD NOT BE FORMED. (DUIIERR 1202)`

CAUSE This means the destination name of a catalog file could not be formed.

ACTION Submit an SR explaining the problem.

1204 `*** THE DUI CATALOG COULD NOT BE ACCESSED. (DUIIERR 1204)`

CAUSE This means a message cannot be read from the DUI's catalog.

ACTION Submit an SR explaining the problem.

1205 `*** THE FILE ! COULD BE OPENED BUT NOT PROPERLY CLOSED. (DUIIERR 1205)`

CAUSE This means a file input by the user exists and was opened but the file header was corrupted or the file could not be closed properly.

ACTION Submit an SR explaining the problem.

1206 `*** THE FILE ! EXISTS BUT IT COULD NOT BE OPENED PROPERLY. (DUIIERR 1206)`

CAUSE The file exists but it could not be opened properly. The protection on the file may be bad.

ACTION Submit an SR explaining the problem.

1209 `*** THE FILE PATH ! COULD NOT BE PARSED OR IS INCOMPLETE. (DUIIERR 1209)`

CAUSE This means the path of the file is invalid for this operating system or it is incomplete.

ACTION Correct the path if it is incorrect or invalid or incomplete. If not. submit an SR explaining the problem.

1212 `*** ! COULD NOT BE COPIED TO !. (DUIIERR 1212)`

CAUSE This is caused when the source cannot be copied to the destination. The source may not exist or can't be accessed. The destination may already exist or the disk space is used up.

ACTION Submit an SR explaining the problem.

1213 `*** THE DIAGNOSTIC SYSTEM FILE ! IS INVALID. (DUIIERR 1213)`

CAUSE This means one of the system files to be added to the diagnostic list file is invalid.

ACTION Submit an SR explaining the problem.

1215 `*** THE DEVICE NAME ! IS TOO LONG. (DUIIERR 1215)`

CAUSE This is caused when a device entered is too long. If it is during an installation then the catalog has a problem.

ACTION Re-enter the device name or fix the catalog.

1218 `*** THE FILE ! DOES NOT EXIST. (DUIIERR 1218)`

CAUSE This means the input file does not exist.

ACTION Input a valid existing file name that follows diagnostic naming rules discussed in the diagnostic development guide. If problems continue, submit an SR explaining the problem.

1219 `*** THE DUI COULD NOT SCAN FOR USER COMMANDS. (DUIIERR 1219)`

CAUSE This means the list of valid program types has been corrupted.

ACTION Submit an SR explaining the problem.

1223 `*** THE FILE PATH ! IS TOO LONG. (DUIIERR 1223)`

CAUSE This means the file path of the input file is too long.

ACTION Input a file with a valid shorter path. If the problems continues, submit an SR explaining the problem.

1224 `*** THE TEMPORARY DIAGNOSTIC LIST FILE CANNOT BE APPENDED TO. (DUIIERR 1224)`

CAUSE This means an internal temporary file cannot be appended to.

ACTION Submit an SR explaining the problem.

1225 `*** THE DIAGNOSTIC DIRECTORY FILE COULD NOT BE CLOSED PROPERLY. (DUIIERR 1225)`

CAUSE This means the actual diagnostic directory file cannot be closed.

ACTION Submit an SR explaining the problem.

1226 `*** THE DIAGNOSTIC LIST FILE COULD NOT BE CLOSED PROPERLY. (DUIIERR 1226)`

CAUSE This means the real diagnostic list file or a temporary one cannot be closed properly.

ACTION Submit an SR explaining the problem.

1227 `*** THE TEMPORARY DIAGNOSTIC DIRECTORY FILE COULD NOT BE CLOSED PROPERLY. (DUIIERR 1227)`

CAUSE This means a temporary diagnostic directory file could not be closed properly.

ACTION Submit an SR explaining the problem.

1228 `*** THE TEMPORARY DIAGNOSTIC LIST FILE COULD NOT BE CLOSED PROPERLY. (DUIIERR 1228)`

CAUSE This means a temporary diagnostic list file could not be closed properly.

ACTION Submit an SR explaining the problem.

1229 `*** THE USER CATALOG COULD NOT BE CLOSED PROPERLY. (DUIIERR 1229)`

CAUSE This means the catalog the user input as the one with the same language as the **DUI** could not be closed properly.

ACTION Submit an SR explaining the problem.

1231 `*** THE DIAGNOSTIC DIRECTORY FILE COULD NOT BE CREATED. (DUIIERR 1231)`

CAUSE This means the actual diagnostic directory file could not be created.

ACTION Check if there is any disk space left.

1232 `*** THE DIAGNOSTIC LIST FILE ! COULD NOT BE CREATED. (DUIIERR 1232)`

CAUSE This means the actual diagnostic list file could not be created.

ACTION Check to see if there is any disk space left.

1233 `*** THE DIAGNOSTIC DIRECTORY FILE COULD NOT BE DELETED. (DUIIERR 1233)`

CAUSE This means the old diagnostic directory file could not be deleted.

ACTION Submit an SR explaining the problem.

1234 `*** THE DIAGNOSTIC LIST FILE COULD NOT BE DELETED.`

`(DUIIERR 1234)`

CAUSE This means the old diagnostic list file could not be deleted.

ACTION Submit an SR explaining the problem.

1235 `*** THERE WAS NO DEVICE GIVEN IN THE MESSAGE CATALOG.`

`(DUIIERR 1235)`

CAUSE This means the user catalog had a device message which was empty. It can also be a problem parsing a device name.

ACTION Correct the device name by editing the catalog. If the problem continues, submit an SR explaining the problem.

1236 `*** THE DESTINATION FILE PATH COULD NOT BE CREATED.`

`(DUIIERR 1236)`

CAUSE This means the destination file path of a catalog, program, or downloadable file could not be parsed.

ACTION Submit an SR explaining the problem.

1237 `*** THE DIAGNOSTIC DIRECTORY FILE COULD NOT BE OPENED.`

`(DUIIERR 1237)`

CAUSE The actual diagnostic directory file could not be opened.

ACTION Submit an SR explaining the problem.

1238 `*** THE DIAGNOSTIC LIST FILE COULD NOT BE OPENED.`

`(DUIIERR 1238)`

CAUSE The actual diagnostic list file could not be opened.

ACTION Submit an SR explaining the problem.

1239 `*** THE TEMPORARY DIAGNOSTIC DIRECTORY COULD NOT BE OPENED. (DUIIERR 1239)`

CAUSE A temporary diagnostic directory file could not be opened.

ACTION Submit an SR explaining the problem.

1240 `*** THE TEMPORARY DIAGNOSTIC LIST FILE COULD NOT BE OPENED. (DUIIERR 1240)`

CAUSE A temporary diagnostic list file could not be opened.

ACTION Submit an SR explaining the problem.

1241 `*** THE USER CATALOG COULD NOT BE OPENED. (DUIIERR 1241)`

CAUSE This means the user catalog matching the language of the `DUI` could not be opened.

ACTION Submit an SR explaining the problem.

1242 `*** THE TEMPORARY DIAGNOSTIC DIRECTORY FILE COULD NOT BE READ. (DUIIERR 1242)`

CAUSE A temporary diagnostic directory file could not be read from.

ACTION Submit an SR explaining the problem.

1243 `*** THE USER CATALOG COULD NOT BE READ. (DUIIERR 1243)`

CAUSE This means the user catalog matching the language of the `DUI` could not be read.

ACTION Submit an SR explaining the problem.

1245 `*** THE TEMPORARY DIAGNOSTIC DIRECTORY COULD NOT BE WRITTEN TO. (DUIIERR 1245)`

CAUSE The temporary diagnostic directory file could not be written to.

ACTION Submit an SR explaining the problem.

1246 `*** ONE OF !'S DEVICE OPTIONS IS INVALID. (DUIIERR 1246)`

CAUSE This means that a device had a option that was invalid or a duplicate of one of its existing options. Valid options are AUTO and DECODE.

ACTION If this happened interactively re-enter the device and its correct options. If this happened during an install then the catalog needs to be fixed.

1247 `*** THE FILE PATH ! IS TOO LONG. (DUIIERR 1247)`

CAUSE This means the path to this file is too long.

ACTION Re-enter a shorter path.

1250 `*** THE DUI AND DIAGNOSTIC DIRECTORY FILE ARE INCOMPATIBLE. (DUIIERR 1250)`

CAUSE This means the version of the DUI used to create the current on-line diagnostic system is not the version of the current DUI.

ACTION Re-install the on-line diagnostic system from the install tape and add in any changes you have made.

1251 `*** THE TEMPORARY DIAGNOSTIC DIRECTORY COULD NOT BE DELETED. (DUIIERR 1251)`

CAUSE The temporary diagnostic directory file could not be deleted.

ACTION Submit an SR explaining the problem.

1252 *** THE TEMPORARY DIAGNOSTIC LIST FILE COULD NOT BE DELETED. (DUIIERR 1252)

CAUSE A temporary diagnostic list file could not be deleted.

ACTION Submit an SR explaining the problem.

1253 *** THE DEFAULT SYSTEM PATH COULD NOT BE FORMED. (DUIIERR 1253)

CAUSE This means the default on-line diagnostic system location could not be parsed correctly.

ACTION Submit an SR explaining the problem.

1254 *** THE FILE ADDITION LIST FILE COULD NOT BE OPENED. (DUIIERR 1254)

CAUSE A temporary diagnostic add file could not be opened.

ACTION Submit an SR explaining the problem.

1255 *** THE FILE ADDITION LIST FILE COULD NOT BE READ. (DUIIERR 1255)

CAUSE A temporary diagnostic add file could not be read from.

ACTION Submit an SR explaining the problem.

1256 *** THE FILE ADDITION LIST FILE COULD NOT BE WRITTEN TO. (DUIIERR 1256)

CAUSE A temporary diagnostic add file could not be written to.

ACTION Submit an SR explaining the problem.

1257 `*** THE FILE ADDITION LIST FILE COULD NOT BE CLOSED. (DUIIERR 1257)`

CAUSE A temporary diagnostic add file could not be closed.

ACTION Submit an SR explaining the problem.

1258 `*** CAN'T FORM THE TEMPORARY DIAGNOSTIC DIRECTORY FILE PATH. (DUIIERR 1258)`

CAUSE This means the file path for a temporary on-line diagnostic system file could not be parsed correctly.

ACTION Submit an SR explaining the problem.

1259 `*** CAN'T FORM THE TEMPORARY DIAGNOSTIC LIST FILE PATH. (DUIIERR 1259)`

CAUSE This means the file path for a temporary on-line diagnostic system file could not be parsed correctly.

ACTION Submit an SR explaining the problem.

1260 `*** CAN'T FORM THE TEMPORARY DIAGNOSTIC PURGELIST FILE PATH (DUIIERR 1260)`

CAUSE This means the file path for a temporary on-line diagnostic system file could not be parsed correctly.

ACTION Submit an SR explaining the problem.

1261 `*** CAN'T FORM THE TEMPORARY DIAGNOSTIC ADDLIST FILE PATH. (DUIIERR 1261)`

CAUSE This means the file path for a temporary on-line diagnostic system file could not be parsed correctly.

ACTION Submit an SR explaining the problem.

1262 `*** THE FILE ADDITION LIST FILE COULD NOT BE DELETED.`
`(DUIIERR 1262)`

CAUSE A temporary diagnostic add file could not be deleted.

ACTION Submit an SR explaining the problem.

1263 `*** THE FILE ADDITION LIST FILE COULD NOT BE CREATED.`
`(DUIIERR 1263)`

CAUSE A temporary diagnostic add file could not be created.

ACTION Check to see if there is disk space available.

1264 `*** A FATAL DUI ERROR HAS OCCURRED ... ABORTING.`
`(DUIIERR 1264)`

CAUSE This means an internal error occurred in print_dui_message while attempting to get a reply from the user.

ACTION Submit an SR explaining the problem.

1265 `COULD NOT RECEIVE MONITOR REPLY FOR A SEMAPHORE REQUEST (DUIERR 1265)`

CAUSE The `DUI` could not receive an ipc message from the diagnostic monitor.

ACTION Verify the diagnostic system is running correctly. If so, submit an SR explaining the problem.

1266 `COULD NOT SEND A SEMAPHORE REQUEST TO THE MONITOR (DUIERR 1266)`

CAUSE The `DUI` could not send a message to the diagnostic monitor

ACTION Verify the diagnostic system is running correctly. If so, submit an SR explaining the problem.

1267 `COULD NOT GET THE MONITOR PORT DURING A SEMAPHORE REQUEST (DUIERR 1267)`

CAUSE The `DUI` could not get the diagnostic monitor's port number.

ACTION Verify the diagnostic system is running correctly. If so, submit an SR explaining the problem.

4001 `*** COULD NOT OPEN FILE ! (DUIERR 4001)`

CAUSE
The **DUI** could not open a usefile for reading or an outfile for writing. The specific cause of the failure is not known.

ACTION
Use the SETVAR TRACE ERROR command. Repeat the command which resulted in the error and note the additional errors which will appear. An easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Unfortunately, OUTFILE and HARDCOPY might not work in this case. Submit an SR against the **DUI** noting all of the errors.

4002 `*** COULD NOT READ FILE ! (DUIERR 4002)`

CAUSE
Could not read from a usefile or infile which had previously been opened successfully for reading. The specific cause of the error is not known.

ACTION
First, check the file to see if there is anything obviously wrong with it (such as being in binary rather than ASCII). If the file seems to be correct use the SETVAR TRACE ERROR command. Repeat the command which resulted in the error appearing noting the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the **DUI** noting all of the errors.

4003 `*** COULD NOT CLOSE FILE ! (DUIERR 4003)`

CAUSE
Failed to close an infile, outfile, or usefile which had previously been opened successfully. The specific reason for the failure is not known.

ACTION
Check to see if the file still exists - it might have somehow been purged from the system after being opened.

If the file still exists then use the SETVAR TRACE ERROR command. Repeat the command which resulted in the error appearing using a different infile, outfile, or usefile if necessary. Note the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called - OUTFILE might not work if the original problem involved an outfile). Submit an SR against the **DUI** noting all of the errors.

4004 `*** FILE ! DOES NOT EXIST (DUIERR 4004)`

CAUSE
Could not open a file because it did not exist.

ACTION
Create the file.

4005 `*** SECURITY VIOLATION ON FILE ! (DUIERR 4005)`

CAUSE
Could not open an outfile or usefile because the user does not have permission to read or write to it.

ACTION
Either change the permissions on the file, log in as a user with a higher security level, or have the system administrator increase your security capabilities.

4009	`*** ACCESS TO THE FILE CANNOT BE GRANTED BECAUSE IT IS CURRENTLY OPENED EXCLUSIVELY. (DUIERR 4009)`
CAUSE	Either could not open or could not read a file because another user or process already has opened the file exclusively.
ACTION	Retry the command after the file has been closed.

4010	`*** THE FILE IS NOT CURRENTLY OPEN (DUIERR 4010)`
CAUSE	Could not read from, write to, or close a particular file because the file descriptor the procedure was trying to use was not recognized by the file system.
ACTION	See if the file has already been closed or if it still exists. If the file is open then use the SETVAR TRACE ERROR command. Repeat the command which resulted in the error appearing using a different infile, outfile, usefile, or hardcopy device if necessary. Note the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called - these might not work if the original problem involved an outfile or hardcopy device). Submit an SR against the DUI noting all of the errors.

4012	`*** HARDCOPY ENVIRONMENT ! IS INVALID (DUIERR 4012)`
CAUSE	There is something wrong with the environment specified with the HARDCOPY command or hardcopy parameter to the RUN command.
ACTION	Re-enter the command either specifying a correct environment or not specifying any environment (i.e. use the default environment). The "correct" environment is operating system and device dependent; the diagnostic system has no control over this feature.

4015	`*** USEFILE IS NOT OPEN (DUIERR 4015)`
CAUSE	Either could not read from or could not close a usefile because none were open.
ACTION	See if the usefile has already been closed or has been purged from the system. If neither of these is the case then use the SETVAR TRACE ERROR command. Repeat the command which resulted in the error appearing using a different usefile if necessary. Note the additional errors which will appear (an easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called). Submit an SR against the DUI noting all of the errors.

4017	`*** INVALID COMMAND, ONLY RUN COMMAND AVAILABLE FOR CODETEST (DUIERR 4017)`
CAUSE	Successfully parsed a command line from a codetest file, however, the command was not the run command. The only DUI command allowed within a codetest file is the run command.
ACTION	Remove the command from the codetest file.

4031 `*** FILE ! ALREADY EXISTS (DUIERR 4031)`

CAUSE The OUTFILE command or the RUN command with the outfile parameter was given. But, an outfile could not be created using the name specified for it because a file with that name already exits.

ACTION Redo the command, giving a different name for the outfile which is to be created.

4032 `*** UNABLE TO CREATE THE SPECIFIED OUTFILE (DUIERR 4032)`

CAUSE Could not create the specified outfile for some reason. The actual reason for the failure will be printed immediately before this message.

ACTION Action taken will depend on the reason for the failure; see the cause/action statement for the message which will immediately precede this one.

5009 `*** THE DUI RECEIVED A NEGATIVE OR ZERO VALUE FOR REDO STACK SIZE (DUIDOREDO 5009)`

CAUSE The REDOSAVE or REDOSIZE command was given with a negative redo stack size specified. The redo stack size must be a value greater than zero.

ACTION Repeat the command, giving a positive integer value for the size of the history stack.

5021 `*** THE DUI TRIED TO PLACE A COMMAND STRING INTO THE SCANNER INPUT BUFFER, BUT ENCOUNTERED AN ERROR. (DUIDOREDO 5021)`

CAUSE The DO or REDO command was given and the command to be repeated was successfully gotten from the history stack. However, the attempt to place the command in a buffer and point to the first character in the command so that it could be parsed failed. This is a particularly strange error which should never occur.

ACTION Use the SETVAR TRACE ERROR command. Repeat the command which resulted in the error and note the additional errors which will appear. An easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Submit an SR against the DUI noting all of the errors.

5022	`*** THE DUI TRIED TO LOCATE THE NUMERIC COMMAND ID ON THE COMMAND HISTORY STACK, BUT THE NUMBER WAS NOT THERE. (DUIDOREDO 5022)`
CAUSE	The DO or REDO command was given with a numeric command identifier, but no matching command could be found in the history stack.
ACTION	Use the LISTREDO command to see the diagnostic system history stack. Then re-enter the command, giving a correct command identifier. Information about command identifiers may be found by using the HELP DO or HELP REDO commands.

5023	`*** THE DUI TRIED TO LOCATE THE STRING COMMAND ID ON THE COMMAND HISTORY STACK, BUT THE STRING WAS NOT THERE. (DUIDOREDO 5023)`
CAUSE	The DO or REDO command was given with a command identifier string, but no matching command could be found in the history stack.
ACTION	Use the LISTREDO command to see the diagnostic system history stack. Then re-enter the command, giving a correct command identifier. Information about command identifiers may be found by using the HELP DO or HELP REDO commands.

5024	`*** THE DUI TRIED TO SAVE THE COMMAND HISTORY STACK, BUT THERE WAS NOTHING ON THE STACK. (DUIDOREDO 5024)`
CAUSE	The REDOSAVE command was given, but the diagnostic history stack could not be saved because it was empty.
ACTION	If the REDOSAVE command is the first command given during the diagnostic session the history stack should be empty - nothing can or should be done to save the stack. If some commands have preceded the REDOSAVE command, then an SR should be submitted against the DUI. Use the SETVAR TRACE ERROR command. Repeat the REDOSAVE command noting the additional errors which will appear. An easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Submit the SR against the DUI noting all of the errors.

6006	`*** INVALID PARAMETERS WERE PASSED. (DUIERR 6006)`
CAUSE	The MODE command was given with both the "multi" and the "single" parameters.
ACTION	Repeat the MODE command using either the "multi" or the "single" parameters, but not both.

6008 `*** LEVEL 0 PRIVILEGE IS NEEDED TO CHANGE THE MODE. (DUIERR 6008)`

CAUSE The MODE SINGLE command was given while the system was in multi-user mode. The user has too low a security capability (less than diag_level_0) for the request to be honored.

ACTION Log in as a user with a higher security level, or have the system administrator increase your security capabilities.

6015 `*** THE SECURITY LEVEL NUMBER WAS INVALID. (DUIERR 6015)`

CAUSE The MODE SINGLE command was given by a user with too low a security level for the request to be honored. The DUI was prevented from telling the user what his/her security level was because the security level could not be translated into an ASCII printable string.

ACTION The actual cause of the failure will be printed immediately before this message; see the cause/action statement for that message. However, if that proves unsatisfactory then invoke the SETVAR TRACE ERROR command. Repeat the MODE SINGLE command noting the additional errors which will appear. An easy way to get a copy of these would be by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Submit the SR against the DUI noting all of the errors.

---

801 `*** COULD NOT LAUNCH COMMAND INTERPRETER (DUIERR 801)`

CAUSE The CI command was given, but the DUI was unable to invoke the command interpreter.

ACTION Invoke the SETVAR TRACE ERROR command. Repeat the CI command and note the additional errors which will appear. An easy way to get a copy of these is by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Submit the SR against the DUI noting all of the errors.

9001 `*** ONLY TEST NUMBERS GREATER THAN ZERO ARE ALLOWED (DUICTERR 9001)`

CAUSE The CODETEST command was given with a "0" or a negative number explicitly specified as a test number to be processed. Only positive integer values greater than 0 may be used as test numbers.

ACTION Repeat the CODETEST command using correct test numbers. If a CODETEST test script contains tests whose numbers are not integer values greater than 0, correct the script.

9002 `*** INVALID TEST RANGE - THE LAST NUMBER IN A RANGE MUST BE GREATER THAN THE FIRST (DUICTERR 9002)`

CAUSE The CODETEST command was given with a test number range ("n/n"), but the last number in the range was smaller than the first.

ACTION Repeat the command correcting the test range so that the second number in the range is greater than the first.

9501 `*** UNABLE TO CONVERT THE SPECIFIED LDEV (!) TO ITS ASSOCIATED PDEV. (DUIERR 9501)`

CAUSE The UNLOCK command was given with an ldev parameter, but the diagnostic system could not convert the specified ldev into its corresponding pdev. Pdevs rather than ldevs are used internally in the diagnostic system so the failure to translate the given ldev into a pdev precludes unlocking the device.

ACTION Repeat the command using a pdev parameter rather than the ldev parameter.

9502 `*** UNABLE TO UNLOCK PDEV (!). (DUIERR 9502)`

CAUSE The UNLOCK command was given. The DUI received a garbled handshaking reply from the underlying process which should have performed the "unlock." Because the handshake was garbled, the DUI is assuming that the device was not unlocked.

ACTION Run something which uses the device (such as its diagnostic) to see if it is really locked or not. Submit an SR against the DUI. If the device is actually still locked the SR is more urgently needed than if it is not.

Invoke the SETVAR TRACE ERROR command. Repeat the UNLOCK command and note the additional errors which will appear. An easy way to get a copy of these is by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Submit the SR against the DUI noting all of the errors.

10000 `*** THE FOLLOWING MESSAGE FROM THE DIAGNOSTIC PROGRAM'S CATALOG CONTAINS A SYNTAX ERROR (DUIERR 10000)`

CAUSE When the user enters the RUN command, the **DUI** obtains information concerning the sections and steps of the diagnostic from the diagnostic's message catalog file. The **DUI** detected a syntax error during the parsing of these section/step messages. This error message will be followed by a copy of the line from the catalog in which the syntax error was detected along with a detailed message describing the problem.

ACTION Submit an SR against the diagnostic including all the information given in this message and the two following.

The RUN command may also be redone naming the sections in some alternate way. That is, if the original command used number ranges try listing the numbers explicitly. If the original command used mnemonics, try the numbers instead. If the original command relied on the defaults (i.e. didn't specify numbers or mnemonics), explicitly name the sections/steps to be run. The diagnostic catalog section/step messages which the **DUI** must parse in order to determine which sections/steps to run are dependent on the form of the input used in the RUN command. By changing the form of input, it is possible that the message with the error can be avoided. An SR should still be submitted though even if this works.

10001 `*** DUI WAS NOT ABLE TO READ MESSAGE 3 OF SET 2 FROM THE DIAGNOSTIC PROGRAM'S CATALOG FILE (DUIERR 10001)`

CAUSE The RUN command was invoked. The **DUI** was not able to read the above mentioned message from the diagnostic program's message catalog. This message contains a list of the default sections for the diagnostic and is needed to determine which sections to run if none were named with the RUN command. An error message indicating the specific problem will follow this message.

ACTION Repeat the RUN command explicitly naming the sections to be run. A list of these may be seen by using the HELP <diagnostic> SECTIONS command.

Additional action may be needed depending on the error message which will follow this one. For example, if that message indicates that the set 2, message 3 message does not exist then an SR should be entered against the diagnostic.

10002 `*** DUI WAS NOT ABLE TO READ MESSAGE 2 OF SET 2 FROM THE DIAGNOSTIC PROGRAM'S CATALOG FILE (DUIERR 10002)`

CAUSE The RUN command was invoked. The **DUI** was not able to read the above mentioned message from the diagnostic program's message catalog. This message contains a list of the default steps for every section in the diagnostic and is needed to determine which steps to run if some steps were not explicitly named within the RUN command. An error message indicating the specific problem will follow this message.

ACTION Repeat the RUN command explicitly naming all steps to be executed. A list of these may be seen by using the HELP <diagnostic> SECTIONS command.

Additional action may be needed depending on the error message which will follow this one. For example, if that message indicates that the set 2, message 2 message does not exist, then an SR should be entered against the diagnostic.

10003 `*** MESSAGE 3 OF SET 2 FROM THE DIAGNOSTIC PROGRAM'S CATALOG IS A BLANK LINE (DUIERR 10003)`

CAUSE The RUN command was invoked on a diagnostic which contains sections, but no sections were named. The **DUI** found that the diagnostic catalog message which should have had a list of the default sections for the diagnostic was a blank line. This is not valid. The **DUI** cannot determine which sections to run in this case.

ACTION Repeat the RUN command explicitly naming all the sections to be executed. A list of these may be seen by using the HELP <diagnostic> SECTIONS command.

An SR should also be entered against the diagnostic.

10004 `*** MESSAGE 2 OF SET 2 FROM THE DIAGNOSTIC PROGRAM'S CATALOG IS A BLANK LINE (DUIERR 10004)`

CAUSE The **DUI** found that the above mentioned message was a blank line. This is not valid. The **DUI** is not able to validate the section and step numbers entered by the user without this information.

The RUN command was invoked on a diagnostic which contains sections and steps, but either no sections were named or some section was named without specifying steps. The **DUI** found that the diagnostic catalog message which should have had a list of the default steps for each section in the diagnostic was a blank line. This is not valid. The **DUI** cannot determine which steps to run in this case.

ACTION Repeat the RUN command explicitly naming all the sections and steps to be executed. A list of these may be seen by using the HELP <diagnostic> SECTIONS command.

An SR should also be entered against the diagnostic.

11010 `*** ERROR - COULD NOT PERFORM REQUESTED OPERATION ON THE SPECIFIED DIAGNOSTIC SYSTEM PROCESS (DUISYSERR 11010)`

CAUSE

1) The DIAGSYSTEM SHOWACTIVE command was given. The proper response should have been for the DUI to print out a list of active diagnostic system processes and their program identifiers. However, one of the identifiers could not be printed because it could not be converted into an ASCII printable string.

2) Either the DIAGSYSTEM RUN <program name> or the DIAGSYSTEM ABORT <program identifier> command was given. The DUI received a handshaking reply from the underlying process which should have performed the "run" or "abort" stating that the action was not completed successfully. The actual reason for the failure will be printed immediately before this message.

ACTION

If the DIAGSYSTEM SHOWACTIVE command was given invoke the SETVAR TRACE ERROR command. Repeat the DIAGSYSTEM SHOWACTIVE command and note the additional errors which will appear. An easy way to get a copy of these is by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Submit the SR against the DUI noting all of the errors.

If the DIAGSYSTEM RUN <program name> or DIAGSYSTEM ABORT <program identifier> command was used then follow the action specified for the message which will immediately precede this one.

11011 `*** ERROR - COULD NOT ABORT THE SPECIFIED DIAGNOSTIC SYSTEM PROCESS (DUISYSERR 11011)`

CAUSE

The DIAGSYSTEM ABORT <program identifier> command was given. The DUI received a garbled handshaking reply from the underlying process which should have performed the "abort." Because the handshake was garbled, the DUI assumes that the process was not aborted.

ACTION

Use the CI command and whatever process status command works on the current operating system to see if the process has actual been aborted. The DIAGSYSTEM SHOWACTIVE command might also work, but its results could be suspect in the current case.

Submit an SR against the DUI. If the process was not actually aborted the SR is more urgently needed than if it was.

Invoke the SETVAR TRACE ERROR command. Repeat the DIAGSYSTEM ABORT <program identifier> command and note the additional errors which will appear. An easy way to get a copy of these is by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Give a complete list of the resultant errors in the SR submitted against the DUI.

11012 *** ERROR - COULD NOT RUN THE SPECIFIED DIAGNOSTIC SYSTEM PROCESS (DUISYSERR 11012)

CAUSE The DIAGSYSTEM RUN <program name> command was given. The DUI received a garbled handshaking reply from the underlying process which should have performed the "run." Because the handshake was garbled, the DUI assumes that the process was not launched.

ACTION Use the CI command and whatever process status command works on the current operating system to see if the process is running. The DIAGSYSTEM SHOWACTIVE command might also work.

Submit an SR against the DUI. If the process was not actually launched the SR is more urgently needed than if it was.

Invoke the SETVAR TRACE ERROR command. Repeat the DIAGSYSTEM RUN <program name> command and note the additional errors which will appear. An easy way to get a copy of these is by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Give a complete list of the resultant errors in the SR submitted against the DUI. It might be necessary to abort the diagnostic system process before repeating the DIAGSYSTEM RUN <program name> command.

11013 *** ERROR - CANNOT LOCATE THE LIST OF DIAGNOSTIC SYSTEM PROCESSES (DUISYSERR 11013)

CAUSE The DIAGSYSTEM SHOWACTIVE command was given. The DUI could not find the diagnostic system table which contains all the information about diagnostic system processes.

ACTION Invoke the SETVAR TRACE ERROR command. Repeat the DIAGSYSTEM SHOWACTIVE command and note the additional errors which will appear. An easy way to get a copy of these is by using the OUTFILE and/or HARDCOPY commands - preferably before the SETVAR command is called. Submit an SR against the DUI giving a complete list of the errors.

3

CIO Channel Adapter Diagnostic

Introduction

The CIO Channel Adapter Diagnostic (CADIAG) is a Diagnostic subsystem program that provides the user with the ability to assess the CIO Channel Adapter functionality. The diagnostic will run under MPE/iX on an HP Precision Architecture RISC computer system from any system terminal. The diagnostic has no interactive commands, but the user can specify which sections and steps are to be run. The user can set parameters to control the handling of error messages and to select the number of test executions to be run. The CIO Channel Adapter diagnostic can also be invoked by the I/O system on catastrophic errors for auto-diagnostic purposes. CIO Channel Adapter functionality is restored by replacing the CIO Channel Adapter PCA which is a Field Replacable Unit (FRU).

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10012.

Auto-Diagnostics

The Channel Adapter diagnostic program can be invoked, by the I/O system on catastrophic errors, for auto-diagnostic purposes. In auto-diagnostic mode, the CADIAG program will execute the following section and steps:

Section 3	Identify
Section 5	Selftest
Section 6	Status
Section 9	Rollcall

Minimum Configuration

The minimum configuration required to run this diagnostic consists of an HP Precision Architecture RISC computer system up and running on the MPE/iX operating system.

Operating Instructions

There is no security level check mechanism within CADIAG. The DUI checks the user's security level before initiating CADIAG. Refer to the section on DUI for a detailed description of user capabilities.

Default Tests

If the user does not specify the sections to be run, the following default sections will be executed:

Section 3	Identify
Section 5	Selftest
Section 6	Status
Section 8	Description

RUN Command

To bring up Online Diagnostics, enter the following command to the MPE/iX prompt:

```
SYSDIAG
```

The system responds with the following prompt indicating that access has been gained to the Online Diagnostics:

```
DUI >
```

Typing HELP causes a summary of the DUI and its commands to be printed. Refer to the DUI Section for details.

Note The device to be tested must be powered up and on line. Device physical locations (pdev) shown in the RUN commands are those of the devices on the "typical A1002A" system configuration described in the chapter on DUI. The pdev value entered must be correct for the system being tested.

To run the diagnostic, you might enter:

```
DUI > RUN CADIAG pdev=4 <RUN Command Options>
                      |         |
                      |    none required for
                      |    default test suite
                      |
          insert physical location of
          channel adapter to be tested here;
          alternatively, type the ldev number
```

All of the RUN command options are used by the CADIAG. A detailed description can be obtained by referring to the section on DUI.

Test Execution

When executed, the diagnostic displays a header and welcome message.

Following the header, CADIAG will attempt to access the channel adapter that was specified in the RUN command. If the identify is not recognized by CADIAG, the diagnostic terminates and the following message is displayed:

```
THE IDENTIFY FUNCTION OF CHANNEL ADAPTER FAILED (CADERR 5604)
```

If at any time, the number of errors generated reaches the limit specified by the user in the ERRCOUNT parameter of the RUN command, the following message will be displayed:

```
More errors encountered than specified in the errcount.
```

The diagnostic will terminate execution immediately upon displaying the above message.

If the ERRPAUSE parameter of the RUN command was assigned a value of "on", then the diagnostic steps after each error is generated and asks the user whether the test should continue:

```
CONTINUE (YES/NO) [ Y ]?
```

If the response is "Y" then the test will be resumed, and if the reponse is "N", the diagnostic will terminate. If the user enters a (Return) the diagnostic defaults to "Y". When the loopcount has completed, the diagnostic terminates and the following message is displayed:

```
CADIAG Exiting . . .
```

If the diagnostic terminated prematurely due to ERRPAUSE or exceeding ERRCOUNT, the above message will not be displayed. Control will return to the DUI.

```
DUI >
```

Detailed Test Descriptions

The following sections are available to run from CADIAG:

Section 3	Identify
Section 5	Selftest
Section 6	Status
Section 8	Description
Section 9	Rollcall
Section 10	Subchannel Status

A description of each section and step will be given, along with the expected output from that section and step assuming no errors have occurred.

Section 3 - IDENTIFY

This section displays the HP product number for the channel adapter selected. The following is executed:

```
Section 3 - IDENTIFY
   The identify function for the channel adapter was completed
   successfully.
 End of Section 3 - IDENTIFY
```

Section 5 -SELFTEST

This section invokes the channel adapter's on-board selftests (Series 3000/950 Only).

```
Section 5 - SELFTEST
   CHANNEL ADAPTER Selftests have completed
      { selftest result messages }
 End of Section 5 - SELFTEST
```

Section 6 - DEVICE STATUS

This section obtains and decodes the device status from the channel adapter.

```
Section 6 - DEVICE STATUS
   The status of the channel adapter is:
        { status messages }
End of Section 6 - DEVICE STATUS
```

Section 8 - DESCRIPTION

This section issues a description command to the channel adapter. The model revision. SPA capability, module type, and serial number from IODC ROM will be returned.

```
Section 8 - DESCRIPTION
   The CA hardware model is n.
    { model 4 = NMOS version, model 8 = TTL version }
   The CA hardware model revision is nnn
   The CA Soft Physical Address capability is nnn
   The CA Software module type is nnn
   The CA Software revision number is n.
   The CA model number is nnn
   The CA serial number is nnn
End of Section 8 - DESCRIPTION
```

Section 9 - ROLLCALL

This section issues a rollcall command to the channel adapter. A bit map will be returned to the caller with the occupied field set to true. The following is executed:

```
Section 9 - ROLLCALL
   Device Adapter Address
                       1 1 1 1 1 1
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   -------------------------------
      <<Rollcall array>>
   A '1' under an address number means that a device adapter was found
   at that address.  A '0' means that no device adapter was found at
   that address.
End of Section 9 - ROLLCALL
```

Section 10 - SUBCHANNEL STATUS

This section issues a command to read and decode the CIO subchannel status.

```
Section 10 -- Subchannel Status
   The Status and Control information for subchannel nnn is:
              { status message }
   { list of messages dependent on status }
   The Subchannel blocksize is nnn
End of Section 10 -- Subchannel Status
```

The status message above is repeated for each subchannel.

Error Messages

The following are general error messages which may be encountered during the execution of CADIAG. System dependent error messages may also be displayed by the diagnostic system; error messages without the trailer (CADERR #) are, in general, generated by the diagnostic system. However, please note that many of the following messages are information messages (and so do not have the error trailer); they are included since they do give information about errors within the channel adapter. The CAUSE/ACTION explanations following each error message below are not displayed to the user.

520	**Channel adapter selftests have FAILED**
CAUSE	One of the on-board selftests failed; the failure was not one from which the channel adapter could recover.
ACTION	Replace the channel adapter card.

526	**An error was detected while running selftests on the channel adapter, but the module is still usable.**
CAUSE	When the selftests were run an error occurred - but the channel adapter was able to recover from it.
ACTION	Ignore the error unless it becomes frequent in which case the channel adapter board should be replaced.

530	**An unrecognized status was returned as the result of running selftests on the channel adapter.**
CAUSE	An unrecognized status was returned from the channel adapter board as a result of running selftests.
ACTION	This message is included for coding completeness but should never actually appear; if it does there has probably been a change in the list of possible status return values and the code will need to be modified. Please report the error to support personnel.

2550	**Parity error detected in high data byte.**
CAUSE	The parity of the high data byte coming over the CIO bus is incorrect.
ACTION	Run selftest on the channel adapter. Test the device adapters. The problem could also be caused by a defective channel adapter buffer card or the extender cable between the channel adapter and the channel adapter buffer card. (Note: not all systems have a channel adapter buffer card and extender cable).

2560	**Parity error detected in low data byte.**
CAUSE	The parity of the low data byte coming over the CIO bus is incorrect.
ACTION	Run selftest on the channel adapter. Test the device adapters. The problem could also be caused by a defective channel adapter buffer card or the extender cable between the channel adapter and the channel adapter buffer card. (Note: not all systems have a channel adapter buffer card and extender cable).

2570 **`The CIO bus controlled by this channel has no power at this time.`**

CAUSE There is no power on the CIO bus.

ACTION Check the power supply - if it is functioning correctly the problem may be in the channel adapter buffer card (assuming the system has one). It might be necessary to replace the channel adapter buffer card (rather than test it) to see if the error persists.

2580 **`The CIO bus has lost power in the past.`**

CAUSE The CIO bus lost power in the past. This message will appear if power was lost on the CIO bus at any time since the last channel adapter reset or power up.

ACTION Check the power supply - if it is functioning correctly the problem may be in the channel adapter buffer card (assuming the system has one). It might be necessary to replace the channel adapter buffer card (rather than test it) to see if the error persists.

2600 **`An internal catastrophic channel error has occurred.`**

CAUSE A catastrophic error has occurred within the channel adapter.

ACTION Replace the channel adapter card.

2610 **`The bus receiver gets different data than the channel is driving.`**

CAUSE The on-board loopback selftest failed; something is wrong with the channel adapter board circuitry.

ACTION Replace the channel adapter card.

2620 **`A data parity error has occurred.`**

CAUSE The channel adapter detected a parity error; something may be wrong with the Midbus or NIO bus.

ACTION Test memory; run selftest on the channel adapter. The Midbus or NIO bus bus converters might be defective - test them, if possible, or replace if necessary.

2630 **`A protocol error on the bus has occurred.`**

CAUSE This could be caused by almost any component in the system.

ACTION Diagnose the entire system to whatever extent is possible. In particular, try to test the SPU, memory, and bus converters.

2640 **`No slave responded to an address generated by the channel.`**

CAUSE The channel adapter attempted to address memory and was unable to complete the bus transaction.

ACTION Test memory. The Midbus or NIO bus bus converters might be defective - test them, if possible, or replace if necessary. The problem might also be that the operating system is specifying an invalid memory address for DMA.

2650	**`The ARQ line was asserted on the CIO backplane but no card responded to the ARQ poll.`**
CAUSE	A device adapter asserted ARQ (asynchronous request) but when the channel adapter polled the device adapters no device adapter responded.
ACTION	Test the CIO bus to whatever extent is possible. If the system has a channel adapter buffer card test it and the attached extender cable (there are no direct tests for these components at this time so they might have to be tested by being replaced to see if the behaviour of the system changes). Test the device adapters.
2660	**`The mstat error code is unknown.`**
CAUSE	An unrecognized module error value was read from the on-board hardware status (io_status) register.
ACTION	This message is included for coding completeness but should never actually appear; if it does, there might be a problem with the channel adapter itself - run selftest. If the channel adapter card appears to be working correctly there has probably been a change in the list of possible status return values and the code will need to be modified. Please report the error to support personnel.
2670	**`The mstat error codes are:`**
CAUSE	An error occurred on the channel adapter board itself. The actual error will be printed in the next message.
ACTION	See the action statement associated with the message printed immediately after this one.
2710	**`SSTAT code means AES,LCD,ERT, or unknown RTS code.`**
CAUSE	An asynchronous event sense, logchannel destroy, error trap, or unknown read transparent status code has been received.
ACTION	Except for the unknown RTS code these are normal signals for the channel adapter to process - no action is necessary. Nothing can be done about the unknown RTS code - it is included as a status code for completeness, but is unlikely to ever occur.
2720	**`SSTAT code means parity error on CIO read data.`**
CAUSE	This error can only occur on a system with an ALINK configured in. Either the ALINK is not generating the correct parity or the channel adapter buffer card is garbling the parity the ALINK is generating. (Note: not all systems have a channel adapter buffer card).
ACTION	Test the ALINK card. The problem might also be in then channel adapter buffer card (on those systems which have one) or the extender cable which connects the buffer card to the channel adapter.

2730	`SSTAT code means no MYAD.`
CAUSE	No device adapter completed a handshake with the channel adapter by asserting a 'my address' signal.
ACTION	Test every device adapter connected to the channel adapter (these can be found by performing a rollcall on the channel adapter). If every one of the device adapters appears to be defective and the system contains a channel adapter buffer card the problem is probably in the channel adapter buffer card or its attached extender cable. If testing the channel adapter buffer card is impractical, replace it and retest the device adapters.

2740	`SSTAT code means internal error.`
CAUSE	An unknown error involving a subchannel occurred within the channel adapter board.
ACTION	Replace the channel adapter card.

2750	`SSTAT code means unknown dma command.`
CAUSE	An unknown error involving a subchannel occurred within the channel adapter board.
ACTION	Replace the channel adapter card.

2760	`SSTAT code means srq on inactive subchannel.`
CAUSE	A service request has been received from a subchannel which is not responding (i.e., handshaking).
ACTION	Test all of the device adapters. If the system has a channel adapter buffer card check it and its attached extender cable. If the device adapters and the extender cable are healthy replace the channel adapter buffer card.

2770	`SSTAT code means RTS overrun.`
CAUSE	The channel adapter received more than sixteen bytes of transparent status from a device adapter. The channel adapter can handle a maximum of sixteen bytes of RTS (read transparent status).
ACTION	Test the device adapters connected to the channel adapter. One of the device adapters may have a defective interface chip.

2780	`SSTAT code is unknown.`
CAUSE	An unrecognized status value was read from the on-board subchannel status register.
ACTION	This message is included for coding completeness but should never actually appear; if it does there has probably been a change in the list of possible status return values and the code will need to be modified. Please report the error to support personnel.

5501	`*** UNABLE TO SELECT THE DEVICE (CADERR 5501)`
CAUSE	The diagnostic could not obtain access to the channel adapter; the reason will be stated in a preceding error message.
ACTION	Refer to the action to be taken which is associated with the preceding error message.

5502	`*** UNABLE TO GET INPUT BUFFER (CADERR 5502)`
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5503	`*** UNABLE TO GET OUTPUT BUFFER (CADERR 5503)`
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5504	`*** UNABLE TO GET HDWR STATUS BUFFER (CADERR 5504)`
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5601	`*** UNABLE TO MAKE STRING FROM NUMBER (CADERR 5601)`
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5602	`*** UNABLE TO PULL BITS FROM A BIT STRING (CADERR 5602)`
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5603	`*** THE CHANNEL ADAPTER MODULE TYPE ! IS WRONG (CADERR 5603) VALID MODULE TYPE VALUES ARE 08, 48, 88, AND C8`
CAUSE	The IODC_TYPE byte of the on-board IO_DC_DATA register contained an unexpected value. This may result from a sysgen problem: the actual hardware configuration may not match the operating system's configuration table.
ACTION	Please report the error to support personnel.

5604	`*** THE IDENTIFY FUNCTION OF CHANNEL ADAPTER FAILED (CADERR 5604)`
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.

5606	`*** THE SELFTEST FUNCTION OF CHANNEL ADAPTER FAILED (CADERR 5606)`
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.

5607	`*** THE STATUS FUNCTION OF CHANNEL ADAPTER FAILED (CADERR 5607)`
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.

5609	`*** THE DESCRIPTION FUNCTION OF CHANNEL ADAPTER FAILED (CADERR 5609)`
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.

5610	`*** THE ROLLCALL FUNCTION OF CHANNEL ADAPTER FAILED (CADERR 5610)`
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.

5611	`*** THE MOVE BUFFER FUNCTION FAILED (CADERR 5611)`
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.

6001	`*** AN UNEXPECTED STATUS WAS RECEIVED (CADERR 6001)`
CAUSE	A completely unanticipated failure occurred somewhere. The actual non-successful status which triggered this error message will be printed.
ACTION	Refer to the action associated with the message which will be printed immediately after this one.

4

Memory Diagnostic

Introduction

The MEMDIAG diagnostic is designed to detect general memory device failures in PA-RISC Memory Array cards, PA-RISC Memory Controller cards, and in PA-RISC Memory Controller/Memory Array combination cards. The diagnostic offers a selection of high level tests that will call lower level routines to access the hardware. If an error is encountered, the Field Replaceable Unit (FRU) will be pinpointed and the test will continue to execute, if possible.

The diagnostic is divided into sections that the user can decide whether or not to run. The user can enter the numbers for the test sections to be run as parameters for the run command, used to invoke the diagnostic.

After the diagnostic has cycled through all the memory pages it was able to access, MEMDIAG will terminate, and the user will be returned to the Diagnostic User Interface (DUI).

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10009.

Minimum Configuration

The hardware required to run MEMDIAG is the PA-RISC system to be tested, consisting of the memory controller card and the memory array card, or of the memory controller/memory array combination card, plus any other equipment required to get the operating system up and running.

Operating Instructions

There is no security level check mechanism within MEMDIAG. The DUI checks the user's security level before initiating MEMDIAG. Refer to the section on the DUI for a detailed description of user capabilities.

Default Tests

If the user does not specify any sections or steps to be run, the default sections are executed, based on the diagnostic mode which has been selected by the Online Diagnostic subsystem. (See the Online Diagnostics Overview discussion of diagnostics modes for details.) The default test is:

Section 9 Trouble Tree

RUN Command

To bring up the Online Diagnostic subsystem or DUI, enter the following command to the HP-UX system prompt:

`%sysdiag`

The system responds with the following Diagnostic User Interface prompt:

`DUI >`

Typing `HELP` causes a summary of the DUI and its commands to be printed. Refer to the DUI Section of this manual for details.

To run the diagnostic, enter:

```
DUI >RUN MEMDIAG
             |
```

no parameters required to load test suite (though parameters can be given)

The user can enter `help memdiag section` and see a menu that briefly describes the section and steps available in the diagnostic.

Parameters to the RUN command

At the invocation of the online diagnostics subsystem diagnostic, the user can define several run time parameters. The `loop` parameter is used to control the number of times the selected sections and steps are executed.

For further information on parameters to the run command, refer to the online diagnostics subsystem DUI ERS. All parameters available in the run command are accepted as parameters when executing this diagnostic.

Test Execution

After the user enters the DUI run command, the diagnostic `MEMDIAG` will be invoked. Once the program is started, a header and welcome message will be displayed on the screen.

Messages will be sent to the screen informing the user of the current test being executed and, upon its completion, the results. If the user did not specify any sections or steps, the default sections and steps will be executed. In the case of `MEMDIAG`, the default section is the Trouble Tree (section 9). Each section has default steps set that will be run if the section is selected without specifying any steps. These default steps are listed under their respective sections in this document.

Whenever `MEMDIAG` prompts the user for input, the user may enter "exit" to terminate the `MEMDIAG` diagnostic and jump back into the DUI system environment, or enter "suspend" to temporarily suspend `MEMDIAG` execution. Either the entire word or any number of characters sufficient to uniquely identify the command may be entered.

The user may then perform tasks within the Diagnostic User Interface (DUI) and later resume/abort execution of `MEMDIAG`.

Early Termination

MEMDIAG runs extensive pattern tests that usually take more than a half hour to complete. Because of this, several options are available for exiting the testing sequence before it terminates in a normal fashion.

1. If time is a critical factor in your testing, go into Section 8, User Interactive, and select specific memory ranges to test. This is a more direct approach, and is much more time-efficient: you can test memory pages mapping to a specific FRU.
2. Before any test is invoked, the following prompt will be displayed to ensure that the user is aware of the time these tests take to complete. The user can then choose to continue testing or not:

```
***
***       WARNING
***
***  The section you have selected will run memory tests that
***  take more than half hour to complete (on average).
***
***  If you don't have the time now, go into USER INTERACTIVE (Section 8)
***  and setup your own memory test by selecting parameters that will
***  specifically test certain ranges and, therefore, shorten test duration.
***
***
***  Do you want to continue with this Section now? (y/n) [y]:
***
```

3. The user may enter **exit** or **suspend** at any MEMDIAG prompt; this will either terminate or temporarily suspend execution of MEMDIAG.
4. If a test is in progress, and the user would like to interrupt its execution, a system-dependent sequence of characters can be entered to interrupt the normal execution cycle of MEMDIAG. The interrupt sequences are <CNTL-C> for HP-UX and <CNTL-Y> for MPE/iX.

Detailed Test Descriptions

The following is a detailed discussion of **MEMDIAG** sections and steps. While in the Diagnostic User Interface, the user may view a menu of sections/steps by using the **help memdiag section** command. The menu gives a brief summary of each section and step. The following table lists the sections/steps of this diagnostic:

```
                         MEMDIAG Menu

Section 1 :           Initialize Card

    step 10 :             initialize memory controller(s)
    step 11 :             memory to all ones
    step 12 :             memory to all zeros


Section 2:            Identify

    step 20 :            Configuration Information
    step 21 :             Identify Information


Section 3:            Status


Section 4:            Memory tests (Link and retest BUSY pages)

    step 40 :             address uniqueness/complement
    step 41 :             walking ones/zeros pattern
    step 42 :             alternating ones/zeros pattern
    step 43 :             ALL section 4 patterns


Section 5:            Memory tests (Link and retest BUSY pages)

    step 50 :             address uniqueness/complement
    step 51 :             walking ones/zeros pattern
    step 52 :             checkerboard/complement pattern
    step 53 :             ALL section 5 patterns


Section 6:            Verify/Sweeper\(Not implemented yet)


Section 7:            EDC Logic test\(Not implemented yet)


Section 8:            USER INTERACTIVE


Section 9:            Trouble Tree (Default Test)
```

Section 1 : INITIALIZE CARD

This section will initialize the memory space to all ones or all zeros and clear the syndrome register(s) of all memory controller(s) in the system. Step 12 will write all zeros to memory then read it back and verify proper storage. Step 11 will write all ones to memory then read it back and verify proper storage. Step 10 will clear the memory controller syndrome registers. This section has three steps, as shown below. The user can select any combination of steps.

```
Step 10 :        Initialize memory controller(s)

Step 11 :        Initialize memory to all 1's

Step 12 :        Initialize memory to all 0's
```

POSSIBLE OUTPUT MESSAGES:

```
Section 1  :   INITIALIZE CARD

        STEP 12 - Initialize memory to ZEROS

----------------------------------------------------------------------------------------------------

        The following PAGE RANGES are being tested          ALL ZEROS

----------------------------------------------------------------------------------------------------

    Start - End      Start - End      Start - End      Start - End

    2598 - 2603      2608 - 2613      2618 - 2622      2625 - 2636

    2638 - 2653      2655 - 2720      2722 - 3015      3018 - 3064

    3066 - 3212      3214 - 3344      3346 - 3425      3430 - 3504

    3506 - 3522      3524 - 3527      3529 - 3555      3557 - 3562

    3564 - 3568      3570 - 3593      3596 - 3625      3627 - 3634

                              ...etc...
```

DEFAULT:

Steps 10 and 12.

Section 2 : IDENTIFY

The Identify section displays configuration and model revision information about the memory system currently being tested. This section has two steps:

Step 20 Configuration Information

Step 21 Model Revision Information (Identify)

The Memory Controller table shown below is from a system where all memory is located onboard the memory controller cards only (there are no associated memory array cards).

SAMPLE OUTPUT MESSAGE 1, when memory interleave is off:

```
Section 2  : IDENTIFY

      STEP 20 - Configuration Information
```

Memory Controller #		1	2	3	4	5
Slot Number	=	3	4	5	7	9
Hard Physical Address (hex)	=	fff8c000	fff90000	fff94000	fff9c000	fffa4000
Associated MA Cards	=	0	0	0	0	0
MC chip size (in bits)	=	1048576	1048576	1048576	1048576	1048576
MC memory size (in Mbytes)	=	8	8	8	8	8
Start Page	=	0	4096	8192	12288	16384
End Page	=	4095	8191	12287	16383	20479
Page Range	=	4096	4096	4096	4096	4096

SAMPLE OUTPUT MESSAGE 2, when memory interleave is off:

```
Section 2  : IDENTIFY

      STEP 20 - Configuration Information
```

Memory Controller #		**1**	**2**
Slot Number	=	0	1
Hard Physical Address (hex)	=	fff80000	fff81000
Associated MA cards	=	3	1

```
>>>MEMORY ARRAY cards under control of MC in Slot # 0
>>>WARNING: If a memory array is disabled, then the corresponding start page
and the corresponding end page will be -1.

If a memory controller is disabled, then all the memory arrays
under this memory controller are also disabled.
```

Memory Array #		**1**	**2**	**3**
Slot Number	=	0	1	2
MA chip size (in bits)	=	1048576	1048576	1048576
MA memory size (in Mbytes)	=	16	16	16
Start Page	=	0	-1	8192
End Page	=	8191	-1	16383
Page Range	=	8192	0	8192

```
>>>MEMORY ARRAY cards under control of MC in Slot # 1
>>>WARNING: If a memory array is disabled, then the corresponding start page
and the corresponding end page will be -1.

If a memory controller is disabled, then all the memory arrays
under this memory controller are also disabled.
```

Memory Array #		**1**
Slot Number	=	0
MA chip size (in bits)	=	1048576
MA memory size (in Mbytes)	=	16
Start Page	=	16384
End Page	=	24575
Page Range	=	8192

SAMPLE OUTPUT MESSAGE **3, when memory interleave is on:**

```
Section 2  : IDENTIFY

      STEP 20 - Configuration Information
```

Memory Controller #		**1**	**2**
Slot Number	=	0	1
Hard Physical Address (hex)	=	fff80000	fff81000
Associated MA cards	=	3	2

```
>>>MEMORY ARRAY cards under control of MC in Slot # 0
>>>WARNING: If a memory array is disabled, then the corresponding start page
and the corresponding end page will be -1.
If a memory controller is disabled, then all the memory arrays
under this memory controller are also disabled.
```

Memory Array #		**1**	**2**	**3**
Slot Number	=	0	1	2
MA chip size (in bits)	=	1048576	1048576	1048576
MA memory size (in Mbytes)	=	16	16	16
MA status	=	Enabled	Disabled	Enabled
Memory Interleave is ON.				

```
>>>MEMORY ARRAY cards under control of MC in Slot # 1
>>>WARNING: If a memory array is disabled, then the corresponding start page
and the corresponding end page will be -1.
If a memory controller is disabled, then all the memory arrays
under this memory controller are also disabled.
```

Memory Array #		**1**	**2**
Slot Number	=	0	2
MA chip size (in bits)	=	1048576	1048576
MA memory size (in Mbytes)	=	16	16
MA status	=	Enabled	Enabled
Memory Interleave is ON.			

The Memory Controller tables below show how **MEMDIAG** will display memory configuration for 9000/720-750. 9000/8x7, and 3000/9x7 machines. Memory array cards on these machines are configured in pairs (labelled as 0A. 0B, 1A, 1B, etc.). Each memory array card can be double-sized; it is possible to disable half the memory on each card in a memory array pair which is double-sized. The operating system treats a double-sized memory array card as if it were a two-bank memory array card. and assigns two SPAs to it.

MEMDIAG treats a pair of slot numbers (e.g., 0 and 1) as a virtual memory controller. Each of these virtual memory controllers controls two memory array cards (A and B) per bank. This means that the user will see the same virtual memory controller displayed twice for a two-bank memory array card. A user can locate a memory array card by combining the virtual memory controller number (slot 0, 1, 2, etc.) and the memory array (A or B) together to form a slot number (0A, 0B, 1A, 1B, etc.).

SAMPLE OUTPUT MESSAGE 4, one bank MC/MA on the mother board:

```
Section 2  :    IDENTIFY


STEP 20 - Configuration Information
```

Memory Controller/ROW # on the mother board slot			
Slot Number	=	3A,3B	1A,1B
Hard Physical Address (hex)	=	6	1
Associated Bank Number	=	0	0
Associated SPA (hex)	=	0	1000000
Start Page	=	0	4096
End Page	=	4095	8191
Page Range	=	4096	4096

```
>>> MEMORY ARRAY cards under control of MC in Slot # on the mother board

>>> WARNING:  If a memory array is disabled, then the corresponding start page
              and the corresponding end page will be -1.
              If a memory controller is disabled, then all the memory arrays
              under this memory controller are also disabled.
```

Memory Array/Row #		**0**	**0**
Slot Number	=	3A	3B
MA chip size (in bits)	=	4194304	4194304
MA memory size (in Mbytes)	=	8	8
MA status	=	Enabled	Enabled

```
>>> MEMORY ARRAY cards under control of MC in Slot # on the mother board

>>> WARNING:  If a memory array is disabled, then the corresponding start page
              and the corresponding end page will be -1.
              If a memory controller is disabled, then all the memory arrays
              under this memory controller are also disabled.
```

Memory Array/Row #		**0**	**0**
Slot Number	=	1A	1B
MA chip size (in bits)	=	4194304	4194304
MA memory size (in Mbytes)	=	8	8
MA status	=	Enabled	Enabled

SAMPLE OUTPUT MESSAGE 5, two-bank MC/MA on the mother board:

```
Section 2  :    IDENTIFY


STEP 20 - Configuration Information
```

Memory Controller # on the mother board slot			
Slot Number	=	0A.0B	0A,0B
Hard Physical Address (hex)	=	0	0
Associated Bank Number	=	1	0
Associated SPA (hex)	=	0	400000
Start Page	=	0	1024
End Page	=	1023	2047
Page Range	=	1024	1024

```
>>> MEMORY ARRAY cards under control of MC in Slot # on the mother board

>>> WARNING:  If a memory array is disabled, then the corresponding start page
              and the corresponding end page will be -1.
              If a memory controller is disabled, then all the memory arrays
              under this memory controller are also disabled.
```

Memory Array/Row #		0	0
Slot Number	=	0A	0B
MA chip size (in bits)	=	1048576	1048576
MA memory size (in Mbytes)	=	2	2
MA status	=	Enabled	Enabled

```
>>> MEMORY ARRAY cards under control of MC in Slot # on the mother board

>>> WARNING:  If a memory array is disabled, then the corresponding start page
              and the corresponding end page will be -1.
              If a memory controller is disabled, then all the memory arrays
              under this memory controller are also disabled.
```

Memory Array/Row #		**1**	**1**
Slot Number	=	0A	0B
MA chip size (in bits)	=	1048576	1048576
MA memory size (in Mbytes)	=	2	2
MA status	=	Enabled	Enabled

The Memory Controller tables below show how **MEMDIAG** displays memory configuration for 9000/710 machines.

SAMPLE OUTPUT MESSAGE 6:

```
Section 2  :    IDENTIFY


STEP 20 - Configuration Information
```

Memory Module #		**0**	**1**
Associated MA (hex)	=	4	4
Start Page	=	0	8192
End Page	=	8191	16383
Page Range	=	8192	8192

>>> MEMORY ARRAY cards under control of Memory Module 0

Memory Array/Row #		1	2
MA Slot Number	=	J12	J13
MC Bank	=	A	A
MA chip size (in bits)	=	1048576	1048576
MA memory size (in Mbytes)	=	8	8

>>> MEMORY ARRAY cards under control of Memory Module 0

Memory Array/Row #		1	2
MA Slot Number	=	J14	J15
MC Bank	=	B	B
MA chip size (in bits)	=	1048576	1048576
MA memory size (in Mbytes)	=	8	8

>>> MEMORY ARRAY cards under control of Memory Module 1

Memory Array/Row #		1	2
MA Slot Number	=	J16	J17
MC Bank	=	A	A
MA chip size (in bits)	=	1048576	1048576
MA memory size (in Mbytes)	=	8	8

>>> MEMORY ARRAY cards under control of Memory Module 1

Memory Array/Row #		1	2
MA Slot Number	=	J18	J19
MC Bank	=	B	B
MA chip size (in bits)	=	1048576	1048576
MA memory size (in Mbytes)	=	8	8

Finally, Step 21 will print out an Identify Table for each memory controller in the memory system; this will contain model/revision information for the system, as well as the SPA associated with the memory array card.

SAMPLE OUTPUT MESSAGE 1:

```
STEP 21 - Identify Information
```

IDENTIFY Information for Memory Controller in Slot #0		
Hardware Model (hex)	=	4
Hardware Revision (hex)	=	4
Software Model (hex)	=	9
Software Revision (hex)	=	0
Software Option (hex)	=	0
Associated Bank Number	=	0
Soft Physical Address (hex)	=	0

SAMPLE OUTPUT MESSAGE 2 for 9000/720-750:

```
STEP 21 - Identify Information
```

IDENTIFY Information for Memory Controller in Slot # on the mother board		
Hardware Model (hex)	=	13
Hardware Revision (hex)	=	0
Software Model (hex)	=	9
Software Revision (hex)	=	0
Software Option (hex)	=	0

SAMPLE OUTPUT MESSAGE 3 for 9000/710:

```
STEP 21 - Identify Information
```

IDENTIFY Information for Memory Controller in Bank A		
Hardware Model (hex)	=	16
Hardware Revision (hex)	=	0
Software Model (hex)	=	9
Software Revision (hex)	=	0
Software Option (hex)	=	0
Soft Physical Address (hex)	=	0

DEFAULT: Steps 20 and 21.

Section 3 : STATUS

This section will return status information from *all* the Memory Controller status register(s) associated with the memory system to be tested. This means the error conditions reported in the syndrome registers will be displayed. An example is shown below of a system with two memory controllers.

SAMPLE OUTPUT MESSAGE 1:

```
SECTION 3 - STATUS


---------------------------------------------------------

          Memory Controller #1             Slot A

---------------------------------------------------------

    >>> NO Memory Errors registered.

---------------------------------------------------------

          Memory Controller #2             Slot B

---------------------------------------------------------

    >>> SINGLE BIT ERROR due to CHECK bits




END Section 3
```

SAMPLE OUTPUT MESSAGE 2, one-bank MC/MA on the mother board:

```
SECTION 3 - STATUS


---------------------------------------------------------

        Memory Controller/Row # 0          slot 0(A,B)

---------------------------------------------------------

  >>> NO Memory Errors registered.




END Section 3
```

SAMPLE OUTPUT MESSAGE 3, two-bank MC/MA on the mother board:

SECTION 3 - STATUS

```
--------------------------------------------------------

          Memory Controller/Row #0            Slot 0(A,B)

--------------------------------------------------------

    >>> Memory Controller is DISABLED.

--------------------------------------------------------

          Memory Controller/Row #1            Slot 0(A,B)

--------------------------------------------------------

    >>> NO Memory Errors registered.



  END Section 3
```

SAMPLE OUTPUT MESSAGE 4, for 9000/710

SECTION 3 - STATUS

```
--------------------------------------------------------

          Memory Controller Bank A

--------------------------------------------------------

    >>> NO Memory Errors registered.

--------------------------------------------------------

          Memory Controller Bank B

--------------------------------------------------------

    >>> NO Memory Errors registered.



  END Section 3
```

DEFAULT:

This section has no steps.

Section 4 : MEMORY TEST (Link and retest BUSY pages)

This section pattern tests memory, checking for basic functionality. It is broken down into three steps, where each step is actually running two tests: the specified pattern and its complement pattern. There is also a fourth step (43) that will batch the patterns into one test.

If no steps are requested in the **run** command, this section will default to select step 43, which runs all six tests. The patterns in this section will also be run in the Trouble Tree section.

The difference between running all tests by selecting the section (which defaults to run step 43), or running all tests by selecting steps 40, 41, and 42 is as follows. Running steps 40, 41, and 42 will sweep memory from start to end six times, once for each pattern used. Running the section will sweep memory from start to end only once, writing and reading back all six patterns in the same cycle. This means that:

- The time that it takes to run step 43 is less than the time that it takes to run the section, because step 43 allocates a page only once, instead of six times.
- Running step 43 may not cover as many busy pages as running steps 40, 41, and 42 may.

The diagnostic will cycle through memory requesting memory pages as it goes; if a page is busy, this section will put that page into a link list, instead of waiting one second for the busy page, before it proceeds on to request the next page. The busy pages in the link list will be retested, after the diagnostic cycles through the whole memory. There are two advantages to chaining busy pages together for later testing. First, diagnostic performance is improved, if there are many busy pages. Second, this also means that the chance of getting a busy page the next time is higher, because the waiting time for a busy page is longer.

The tests are explained below.

Step 40 — **Address Uniqueness/Complement test:** This test will write the address value of a memory location into itself. For example, the value 12 will be stored in memory address location 12. This test will verify that every storage cell exists as a separate and unique entity. Once this is verified, the complement of this test will be performed, where the complement of each address will be written into each address location.

Step 41 **Walking Ones/Zeros pattern test:** This test writes a walking one pattern into memory, then it is read back and verified. The diagnostic will then write and verify a complementary walking zero pattern. The Walking Ones pattern test will cycle through memory writing 32 patterns (each of 32 bits) into consecutive memory locations.

The patterns cycled through are as follows:

```
1000 0000 0000 0000 0000 0000 0000 0000
0100 0000 0000 0000 0000 0000 0000 0000

0010 0000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000

                . . . etc . . .

0000 0000 0000 0000 0000 0000 0000 1000
0000 0000 0000 0000 0000 0000 0000 0100

0000 0000 0000 0000 0000 0000 0000 0010
0000 0000 0000 0000 0000 0000 0000 0001
```

Step 42 **Alternating Ones/Zeros pattern test:** This test writes an Alternating Ones pattern into memory and then reads it back and verifies memory contents. Again, the complementary pattern of Alternating Zeros is written and read back upon successful completion of the initial pattern.

The Alternating Ones pattern is as follows :

```
1010 1010 1010 1010 1010 1010 1010 1010
```

Step 43 **All Patterns Section 4:** Starting at Page 0, request pages until end of memory is reached. For each page obtained, write and read back *all* patterns from Section 4 into this page.

POSSIBLE OUTPUT MESSAGES:

```
   Section 4  :   MEMORY TEST   (Link and retest BUSY pages)

        STEP 40 - Address Uniqueness/Complement Test

-----------------------------------------------------------------------------

     The following PAGE RANGES are being tested          ADDRESS UNIQUENESS

-----------------------------------------------------------------------------

     Start - End       Start - End       Start - End       Start - End

     2598 - 2603       2608 - 2613       2618 - 2622       2625 - 2636

     2638 - 2653       2655 - 2720       2722 - 3015       3018 - 3064

     3066 - 3212       3214 - 3344       3346 - 3425       3430 - 3504

     3506 - 3522       3524 - 3527       3529 - 3555       3557 - 3562

     3564 - 3568       3570 - 3593       3596 - 3625       3627 - 3634

     3636 - 3673       3570 - 3593       3596 - 3625       3627 - 3634

                                ...etc...

  END Section 4
```

DEFAULT:

Step 43

Section 5 : MEMORY TEST (Link and retest BUSY Pages)

This section is broken down into three steps. where each step is actually running two tests: the specified pattern and its complement pattern. There is also a fourth step (53) that will batch all the patterns into a single test.

The difference between running all tests by selecting the section (which defaults to run step 53). or running all tests by selecting steps 50. 51. and 52 is as follows. Running steps 50, 51. and 52 will sweep memory from start to end six times, once for each pattern used. Running the section will sweep memory from start to end only once, writing and reading back all six patterns in the same cycle. This means:

- The time that it takes to run step 53 is less than the time that it takes to run the section, because step 53 allocates a page only once, instead of six times.
- Running step 53 may not cover as many busy pages as running steps 50, 51, and 52 may.

The diagnostic will cycle through memory requesting memory pages as it goes; if a page is busy. this section will put the busy page into a link list, instead of waiting 5 seconds for the busy page. before it proceeds on to request the next page. The busy pages in the link list will be retested. after the diagnostic cycles through the whole memory. There are two advantages to chaining busy pages together for later testing. First, diagnostic performance is improved. if there are many busy pages. Second. this also means that the chance of getting a busy page the next time is higher, because the waiting time for a busy page is longer.

The tests are explained below:

Step 50 — **Address Uniqueness/Complement test:** This test will write the address value of a memory location into itself. For example the value 12 will be stored in memory address location 12. This test will verify that every storage cell exists as a separate and unique entity. Once this is verified. the complement of this test will be performed. where the complement of each address will be written into each address location.

Step 51 **Walking Ones/Zeros pattern test:** This test writes a walking one pattern into memory, then it is read back and verified. The diagnostic will then write and verify a complementary walking zero pattern. The Walking Ones pattern test will cycle through memory writing 32 patterns (each of 32 bits) into consecutive memory locations. The patterns cycled through are as follows:

```
1000 0000 0000 0000 0000 0000 0000 0000
0100 0000 0000 0000 0000 0000 0000 0000

0010 0000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000

               . . . etc . . .

0000 0000 0000 0000 0000 0000 0000 1000
0000 0000 0000 0000 0000 0000 0000 0100

0000 0000 0000 0000 0000 0000 0000 0010
0000 0000 0000 0000 0000 0000 0000 0001
```

The diagnostic will then write and verify a complementary walking zero pattern.

Step 52 **Checkerboard pattern/Complement test:** This test will write a checkerboard pattern to the memory space. Afterwards it will read back the memory and verify that the data was stored correctly; then the complement test will be done. The pattern written is the Alternating Ones pattern followed by the complement of this pattern. After each memory address location is written, the pattern is toggled and so on.

Step 53 **All patterns Section 5:** Starting at Page 0, request pages until end of memory is reached. For each page obtained, write and read back *all* patterns in Section 5 into this page.

POSSIBLE OUTPUT MESSAGES:

```
  Section 5  :   MEMORY TEST  (Link and retest BUSY pages)

        STEP 50 - Address Uniqueness/Complement Test

--------------------------------------------------------------------------

     The following PAGE RANGES are being tested          ADDRESS UNIQUENESS

--------------------------------------------------------------------------

      Start - End        Start - End         Start - End        Start - End

      2598 - 2603        2608 - 2613         2618 - 2622        2625 - 2636

      2638 - 2653        2655 - 2720         2722 - 3015        3018 - 3064

      3066 - 3212        3214 - 3344         3346 - 3425        3430 - 3504

      3506 - 3522        3524 - 3527         3529 - 3555        3557 - 3562

      3564 - 3568        3570 - 3593         3596 - 3625        3627 - 3634

      3636 - 3673        3570 - 3593         3596 - 3625        3627 - 3634

      3636 - 3673        3676 - 3679         3681 - 3770        3772 - 3776

                                   ...etc...

  END Section 5
```

DEFAULT:

Step 53

Section 6 : SWEEPER FUNCTION (Unimplemented)

This section runs a basic sweeping function to read all memory locations. If any errors are found, those errors will be logged in an error log for future correction.

A sweeper function will *sweep* through memory from beginning to end and read all memory locations that it can obtain access to. By making a read of a memory location the data from that location will pass through the EDC logic and if there is a SBE error within the data read it will show up in the associated memory controllers status word. The sweeper function will read a memory range and then check the status register to see if an error occurred. If an error is flagged, the sweeper function will log the location where it occurred, clear the error flag and continue the sweep. This function is important because it can help monitor the level of soft errors occurring in memory, this can help reduce the probability of HPMCs (machine crashs) by allowing the user to see possible problem areas in memory. If an area is showing repetitive occurrence of soft errors, then possibly this card should be changed. The type of error, address location, and associated memory controller will be logged in a file for monitoring of soft error levels as a preventative maintenance measure. The diagnostic will display the message shown below at the completion of the sweeping function. The errors logged will be broken down into the categories of *buffer errors* and *card errors*.

This section is currently unimplemented and will be implemented dependent upon Hardware and Operating System changes that are needed to mask any HPMC that could occur as a result of this test. The reason this may cause HPMCs is that by sweeping through memory and reading each location you will be touching areas of memory on a regular basis. If you read a memory location and there is a hard error there (a stuck bit) and a transient error occures during the read (say a glitch) you will have a double bit error which will cause an HPMC on the system. So by regular sweeps through memory you are increasing the probability of causing HPMCs.

POSSIBLE OUTPUT MESSAGES:

```
Section 6  :   SWEEPER FUNCTION TEST

Sweeper function has completed.

TOTAL ERRORS logged ==>> 5
Card   errors :  5
Buffer errors :  0
END Section 6
```

DEFAULT:

```
This section has no steps
```

Section 7 : EDC LOGIC TEST (Unimplemented)

This section runs an error logic test sequence to verify functionality of the EDC logic on board the memory card(s). This test will write false data into the Syndrome Register usually by writing a force_error register that is architected in the system. This force_error register will cause the EDC hardware to flag a data error because the Syndrome will not match up to the data read. The data finally put out on the bus will be corrected by the EDC logic: but it is corrected only *on-the-fly* and not in the actual memory location. This means that the EDC logic can detect a SBE and correct that single bit value before putting it out onto the bus; however, the value stored in memory is still incorrect.

The force_error register name varies from system to system and some may not even have this feature designed in. If that is the case, the diagnostic will print a message stating that the test cannot be performed because it is not functionally supported.

This section is currently unimplemented and will be implemented dependent upon Hardware and Operating System changes that are needed to mask any HPMC that could occur as a result of this test.

POSSIBLE OUTPUT MESSAGES:

```
Section 7  :   EDC LOGIC TEST

EDC Logic Test completed successfully.

END Section 7
```

DEFAULT:

```
This section has no steps
```

Section 8 : USER INTERACTIVE

This section will present the user with a menu presenting different areas requiring user interaction. The user will input the area they wish to proceed in. They will then be prompted for whatever parameters are needed for that area to be performed. This section will be menu driven and grow with the functionality needs of each PA-RISC system that is added to the diagnostic repertoire.

The main menu is explained below along with the prompts that will be displayed with each user selection.

POSSIBLE OUTPUT MESSAGES:

```
   Section 8  :   USER INTERACTIVE

================================================================================
                           MEMDIAG MENU
================================================================================

 1.  Memory Test  (using parameter settings)
 2.  EDC Test     (using parameter settings)
 3.  Sweep Memory (using parameter settings)
 4.  Set MEMDIAG parameters
 5.  View MEMDIAG parameters
 6.  View Configuration Information
 7.  View Page Status Lists
 8.  View Page Status Summary
 9.  View Memory Status (over Range)
10.  HELP Menu
11.  Exit to DUI

Input number >>>
```

If users select *1* from the Main Menu (*Memory Test*), the test will write the selected pattern(s) to all available memory locations, beginning at the starting page and ending at (starting page + page range - 1). This selection will obtain the pages within the page range specified by the user before the test is done, and then will return ownership to the system after the test is performed.

The memory test will run until the End of Range (User Input) or End of Memory is encountered, whichever comes first.

SAMPLE OUTPUT MESSAGE:

```
Section 8 : MEMORY TEST

--------------------------------------------------------------------------------------

      The following PAGE RANGES are being tested              ALL PATTERNS

--------------------------------------------------------------------------------------

     Start - End        Start - End        Start - End        Start - End

     2598 - 2603        2608 - 2613        2618 - 2622        2625 - 2636

     2638 - 2653        2655 - 2720        2722 - 3015        3018 - 3064

     3066 - 3212        3214 - 3344        3346 - 3425        3430 - 3504

     3506 - 3522        3524 - 3527        3529 - 3555        3557 - 3562

     3564 - 3568        3570 - 3593        3596 - 3625        3627 - 3634

     3636 - 3673        3570 - 3593        3596 - 3625        3627 - 3634

     3636 - 3673        3676 - 3679        3681 - 3770        3772 - 3776

                                 ...etc...
```

Remember that you might not be able to obtain all the pages you requested. Many pages are permanently reserved by the operating system.

To get a general idea of the size and location of your own operating system, you can go into selection *7* (*View Page Status Lists*) and view all Reserved Pages.

If users select *2* from the Main Menu (*EDC Test Memory Range*), this function will test EDC (Error Detection and Correction) logic corresponding to the memory locations falling between the starting page and (starting page + page range - 1).

When the data is written to a memory location, it will pass through EDC logic, where a Syndrome word is encoded (7 to 9 bits long) and stored in an EDC RAM. This Syndrome word will be read whenever its associated memory location is read. Both the data word and the Syndrome word will pass back through EDC logic where the Syndrome word will be decoded.

This decoded Syndrome word and the actual data word read from the memory location will be compared with each other. If they are not equal, an error exists. The PA-RISC architecture has special registers that can be written to change the correct Syndrome word to a false one. This will test the EDC logic to see if it detects this error.

This selection will obtain the pages within the page range specified by the user before the test is done, and then returns ownership to the system after the test is performed. Remember that you might not be able to obtain all the pages you requested. Many pages are permanently reserved by the operating system.

If users select *3* from the Main Menu (*Sweep Memory Range*), the function will read all available memory locations, beginning at the starting page and ending at (starting page + page range - 1). The memory contents will scroll on screen. Every eighth memory address will display above a line containing the contents of that address, and the next seven memory locations after that address. The output will be presented in a hexadecimal format for a more compact display. A sample is shown below:

```
PAGE 0
------------------------------------------------------------------------------------------------

Starting address for this row >>     0x 0000 0000

00000000  00000001  00000002  00000003  00000004  00000005  00000006  00000007
00000008  00000009  0000000A  0000000B  0000000C  0000000D  0000000E  0000000F
00000020  00000021  00000022  00000023  00000024  00000025  00000026  00000027
00000028  00000029  0000002A  000000A3  000000A4  000000A5  000000A6  000000A7

Starting address for this row >>     0x 0000 00A8

000000A8  000000A9  000000AA  000000AB  000000AC  000000AD  000000AE  000000AF
000000B0  000000B1  000000B2  000000B3  000000B4  000000B5  000000B6  000000B7
000000B8  000000B9  000000BA  000000BB  000000BC  000000BD  000000BE  000000BF
000000C0  000000C1  000000C2  000000C3  000000C4  000000C5  000000C6  000000C7

                         ... etc ...
```

If the user selects *4* from the Main Menu (*Set MEMDIAG Parameters*), the option to set a variety of MEMDIAG parameters is presented, which permits the user to tailor the diagnostic to more closely conform to the environment in which it will be running:

```
   Set MEMDIAG Parameters

----------------------------------------------------------------------------

      1. Launch Memory Test NOW
      2. View Current Parameter Settings
      3. Change PATTERN parameter
      4. Change START PAGE parameter
      5. Change PAGE RANGE parameter
      6. Change PAGE WAIT parameter
      7. Change ALLOC SIZE parameter
      8. Change LOOP COUNT parameter
      9. Change REPEAT PROMPT? parameter
      10. Map a physical address to MA & MC
      11. Exit to Main Menu
      12. Exit to DUI

   <CR> STEP THROUGH ALL PARAMETER CHOICES

----------------------------------------------------------------------------
```

If the user wants to change all the parameters, he should hit <CR>, which will walk him through each parameter, displaying its current setting and also the possible input to choose from. If only one (or a few) parameters are to change, select the number corresponding to that parameter(s).

The following list explains the *Set Memdiag Parameter Menu* options:

1. Launch Memory Test NOW This selection will launch the memory test with current parameter settings.

2. View Current Parameter Settings This selection will display the Memdiag Parameter table.

3. Change PATTERN Parameter The user will be presented with a Pattern Menu of 13 choices, The pattern selected will be used when running *Memory Test (#1)*.

```
================================================================================
                       MEMDIAG PATTERN MENU
================================================================================

                         1. All Ones
                         2. All Zeros
                         3. Alternating Ones
                         4. Alternating Zeros
                         5. Walking One
                         6. Walking Zero
                         7. Checkerboard
                         8. Checkerboard complement
                         9. Address Uniqueness
                        10. Address Uniqueness complement
                        11. ALL straight patterns
                        12. ALL complement patterns
                        13. ALL patterns listed above
                       <CR> Keep current  setting


                  Input number of pattern desired >>
```

Pattern reading and writing is the time-intensive part of memory testing. By increasing the number of test patterns used, you increase the time it takes the test to complete. It is recommended that you test system performance with ONE pattern, before trying to batch patterns in a memory test. The following warning will print out before you are presented with the pattern menu shown above:

```
***
***     WARNING                              ( MEMWARN 10285 )
***
***     Selecting groups of patterns in the User Interactive section
***     will cause CPU intensive inner loops and system performance
***     will be reduced.
***
***     If you have not already done so, run with ONE pattern to test
***     out performance degradation before trying multiple pattern tests.
***
```

4. Change START PAGE parameter The user will be prompted for a test Starting Page. Memory page mappings will be displayed to allow the user to test a specific memory controller or memory array card.

A sample output is shown below:

```
------------------------------------------------------------------------------------
   Memory Cont/Array in Slot 7 PAGES        0 to  4095      RANGE = 4096

   Memory Cont/Array in Slot 8 PAGES     4096 to  8191      RANGE = 4096

   Memory Cont/Array in Slot 9 PAGES     8192 to 12287      RANGE = 4096
------------------------------------------------------------------------------------
   Currently = 0

   Input STARTING PAGE or <CR>to keep current setting >>>
```

5. Change PAGE RANGE parameter The user will be prompted for a test Page Range. Memory page mappings will be displayed to allow for the user to test a specific memory controller or memory array card.

A sample output is shown below:

```
------------------------------------------------------------------------------------
   Memory Cont/Array in Slot 7 PAGES        0 to  4095      RANGE = 4096

   Memory Cont/Array in Slot 8 PAGES     4096 to  8191      RANGE = 4096

   Memory Cont/Array in Slot 9 PAGES     8192 to 12287      RANGE = 4096
------------------------------------------------------------------------------------
   Currently = 8192

   Input PAGE RANGE or <CR>to keep current setting >>>
```

6. Change PAGE WAIT parameter This is the number of seconds that the program will wait on a BUSY page while trying to obtain access to it, before continuing on to request the next virtual memory page.

7. Change ALLOC SIZE parameter The normal memory tests will cycle through memory requesting ONE page at a time. This is due to an MPE/iX operating system limitation. If you are on HP-UX or in Single User Mode, you will probably want to increase this parameter and run tests from within the User Interactive Section.

Warning

The MPE/iX operating system does not have clean-up routines if MEMDIAG were to fail, meaning if you set ALLOC SIZE = 2048 and MEMDIAG is granted ownership of these pages and then dies, these pages are lost to the system. This is dangerous. The HPUX operating system is able to recover these pages, therefore, this is not a problem for HPUX. It is recommended that you use ONE page for ALLOC SIZE when running MEMDIAG on an MPE/iX system.

8. Change LOOP COUNT parameter The user will input a TEST loop counter for repeat test cycles. Loop Count will be preempted by the Repeat Prompt query.

9. Change REPEAT TEST PROMPT? parameter If ON, prompt the user at the end of the test cycle to repeat last test. If OFF, continue normal diagnostic flow of execution.

The Repeat Prompt Query will take precedence over Loop Count.

All user input parameter values will ONLY affect the TESTS run from WITHIN the User Interactive Section of MEMDIAG (Section 8). Specifically, the test selections affected are selections #1 (*Memory Test*), #2 (*EDC Test*), and #3 (*Memory Sweep*). The User Parameter Table does NOT affect other menu selections.

If the user selects *5* (*View MEMDIAG Parameters*) from the Main Menu, a menu similar to the following will be displayed:

USER PARAMETERS	Selected	Valid Range	Default
PATTERN(S)	All Ones	1 to 13	ALL ONES
START PAGE	100	0 to 20479	0
PAGE RANGE	3000	1 to 20480	20480
PAGE WAIT	5	0 to 10	0
ALLOC SIZE	1024	1 to 2048	1
LOOP COUNT	2	1 to 10000	1
REPEAT PROMPT?	ON	ON/OFF	OFF

Typing a carriage return will leave parameters at the current setting.

If the user selects *6* (*View Configuration Information*) from the Main Menu, ALL configuration information about any Memory Controller and/or Memory Array cards in the system will be displayed. The following is a sample display:

```
------------------------------------------------------------------------------------
  Memory Controller #        1          2          3          4          5
------------------------------------------------------------------------------------
  Slot Number         =      4          8          7          6          5

  Hard Physical       =  fff90000   fffa0000   fff9c000   fff98000   fff94000
  Address (hex)

  Associated          =        0          0          0          0          0
    MA Cards

  MC chip size        =   1048576    1048576    1048576    1048576    1048576
   (in bits)

  MC memory size      =        8          8          8          8          8
   (in Mbytes)

  Start Page          =        0       4096       8192      12288      16384

  End Page            =     4095       8191      12287      16383      20479

  Page Range          =     4096       4096       4096       4096       4096
------------------------------------------------------------------------------------
```

If the user selects 7 (*View Page Listings*) from the Main Menu, a list of the pages in the system that are status pages selected by the user will be printed.

```
========================================================================
                        PAGE STATUS MENU
========================================================================

                         1.  View Reserved   Pages
                         2.  View Available  Pages
                         3.  View Busy       Pages
                         4.  View Hole       Pages
                         5.  View Locked     Pages
                         6.  View Bad        Pages
                         7.  View Bad/Locked Pages
                         8.  View Page Status Summary
                         9.  Exit to Main Menu
                        10.  Exit to DUI

                    Input number >>>
```

A sample Reserved Page List follows:

```
--------------------------------------------------------------------------
                        RESERVED PAGE LIST
--------------------------------------------------------------------------

0      to 3180     4000 to 4200        4210 to 4211        4224 to 5000
5001   to 5009     6001 to 6001        8010 to 8020        9111 to 9111
9200   to 9201     9990 to 9999       10010 to 10100      12000 to 12001
13433 to 13435    13500 to 13555      13599 to 13600      13701 to 13701
                               ... etc ...
--------------------------------------------------------------------------
                    Reserved pages =   4000

                    TOTAL PAGES    =   20479
--------------------------------------------------------------------------
```

Page Status Types are defined under Main Menu selection 9 (*View Memory Status*).

If the user selects *8* (*View Memory Status Summary*) from the Main Menu, a synopsis of the ALL System Page Status will be printed out. A sample output follows:

```
-----------------------------------------------------------------------
                      STATUS SUMMARY SNAPSHOT
-----------------------------------------------------------------------
                      Reserved pages     =   3761
                      Available pages    =  16716
                      Busy pages         =      2
                      Locked pages       =      0
                      Bad pages          =      0
                      Bad/Locked pages   =      0
                      Hole pages         =      0
-----------------------------------------------------------------------
                      TOTAL PAGES        =  20479
```

The Reserved pages consist primarily of operating system space. Some pages may stay constantly busy because they are used for system or program stacks. All these page assignments are dynamic. Even the Reserved pages total may vary to a small extent between different Status Summary Snapshots.

Page Status Types are defined under Main Menu selection *9 (View Memory Status)*.

If the user selects *9* (*View Memory Status Over Range*) from the Main Menu, a starting page and a page range will be prompted for. The Page and Range values input for the Memory Status selection DO NOT change those parameters in the TEST parameter table (set in Main Menu selection #4). This function will check status on all the pages requested, and will present a snapshot of the memory at the time of the request.

The status of the pages will be scrolled on the screen. The output will have a starting page number, followed by the status of that page and of the following seven pages. Viewing memory range does not require that access be obtained to the page in order to view its status; therefore, you may view any page status within the operating system valid range of pages.

The majority of page status types will be Available, Busy, and Reserved. The other four page types are fairly uncommon, and are used for special cases within an operating system.

A sample is presented below:

```
------------------------------------------------------------------------
(ROW)
Start                    PAGE STATUS
Page
------------------------------------------------------------------------

0902      | Reserv  Reserv  Reserv  Reserv  Reserv  Reserv  Reserv  Reserv
0911      | Busy    Busy    Busy    Busy    Busy    Busy    Busy    Avail
0920      | Busy    Busy    Avail   Reserv  Reserv  Avail   Busy    Avail
0929      | Reserv  Avail   Avail   Avail   Avail   Avail   Reserv  Avail
0938      | Reserv  Avail   Avail   Avail   Avail   Avail   Reserv  Avail
0947      | Reserv  Avail   Avail   Avail   Avail   Avail   Reserv  Avail
0956      | Reserv  Avail   Avail   Avail   Avail   Avail   Reserv  Avail
0965      | Locked  Locked  Locked  Bad_Lk  Bad_Lk  Bad_Lk  Reserv  Avail

                        ... etc ...
------------------------------------------------------------------------
```

Reserv This means that the page is permanently reserved by the operating system, and cannot be obtained for testing (in an on-line environment).

Avail This means that the page is NOT LOCKED, and is possibly available for testing because it has not been allocated to any other process. However, many pages may show Available status and not be LOCKABLE due to constant system usage.

Busy This means that the page is currently busy and, therefore, is unavailable for testing until the process currently using it has completed. This is usually associated with I/O.

Hole Page numbers with no physical manifestation are termed "holes". For example, if the maximum page of the system is 20479, and somehow page 20500 was requested, it would not exist and the status of "hole" would be returned.

Locked This means that the page is currently locked by a process (that process may be your own diagnostic or an external process).

Bad This means the page has been marked "bad" by the operating system. For example, if a diagnostic found excessive errors within a page, it could mark the page "bad" and lock out other processes from accessing this corrupted page. This is dependent upon the operating system's ability to bar user access to corrupted pages.

Bad_Locked This means that a page is both Locked and deemed Bad within the system.

If the user selects *10* (*Help Menu*) from the Main Menu, the following menu is displayed:

```
========================================================================
                          MEMDIAG HELP MENU
========================================================================

                      1.   Memory Test  (using parameter settings)
                      2.   EDC Test     (using parameter settings)
                      3.   Sweep Memory (using parameter settings)
                      4.   Set MEMDIAG parameters
                      5.   View MEMDIAG parameters
                      6.   View Configuration Information
                      7.   View Page Status Lists
                      8.   View Page Status Summary
                      9.   View Memory Status (over Range)
                     10.   Exit to Main Menu
                     11.   Exit to DUI

                     Input the area you want >>>
```

Selecting any of these items will scroll the explanations presented in this manual for each of the listed areas.

If the user selects *11* (*Exit to DUI*) from the Main Menu, control is returned to the DUI command line.

Section 9 : TROUBLE TREE (Default Test)

This section can be used to perform all possible diagnostics within the `MEMDIAG` repertoire. This is the **default** section for the Memory Array Diagnostic. If you just type **run memdiag**, then this section will be executed. The testing sequence for this section is as follows:

1. Initialize memory controller status to clear status errors.
2. Print configuration information
3. Perform memory pattern test with
   a. All Zeros
   b. All Ones
   c. Address Uniqueness
   d. Address Uniqueness Complement
   e. Walking Ones
   f. Walking Zeros
   g. Alternating Ones
   h. Alternating Zeros

This test cycles once through memory, instead of cycling eight times as in previous versions. This means that the time it takes to run the section is significantly less than before, but it may not cover as many busy pages as before.

POSSIBLE OUTPUT MESSAGES:

```
------------------------------------------------------------------------------------------------
                                     TROUBLE TREE
------------------------------------------------------------------------------------------------

   >>>  SECTION 9

   >>>  SYSTEM CONFIGURATION TABLE

           (tables displayed here)


   >>>  CONFIGURATION INFORMATION RETRIEVED SUCCESSFULLY



   >>>  The following PAGE RANGES are being tested with DEFAULT PATTERNS:
   >>>  ONES,ADDR_UNIQUE,WALK_ONES,ALTER_ONES and their complements.

     (tested page list displayed here)


   >>>
   >>>
   >>>  MEMDIAG HAS COMPLETED SUCCESSFULLY WITHOUT FINDING ANY MEMORY ERRORS.
   >>>
   >>>


   >>>  END OF SECTION 9
```

DEFAULT:

This section has no steps

Error and Warning Messages

Error messages may be displayed as a result of Syndrome Register reads, or as a result of memory errors occurring during the operation of `MEMDIAG`; memory errors may also generate warning messages.

Syndrome Register Read

One Syndrome Register is associated with each Memory Controller in a memory system. The Syndrome register will contain the status of the last memory read or write operation to memory under the supervision of the associated Memory Controller. The memory operation will result in a SUCCESSFUL completion (Message 10250) or in a FAILURE to complete (Messages 10251 thru 10258).

The possible status values are listed below. These messages are actual decoding of the hardware status returned after the memory transaction.

Cause: Memory hardware failed to complete memory transaction.

Action: Replace chip or entire card indicated in the message immediately following this one that will pinpoint the exact Field Replaceable Unit (FRU) that is malfunctioning.

ERROR MESSAGE 10250	`NO Memory Errors registered.`
ERROR MESSAGE 10251	`SINGLE BIT ERROR due to UNKNOWN cause.`
ERROR MESSAGE 10252	`SINGLE BIT ERROR due to DATA bits.`
ERROR MESSAGE 10253	`SINGLE BIT ERROR due to CHECK bits.`
ERROR MESSAGE 10254	`SINGLE BIT ERROR logged on PARITY read.`

ERROR MESSAGE 10255	SINGLE BIT ERROR logged on PARITY write.
ERROR MESSAGE 10256	MULTI BIT ERROR due to UNKNOWN cause.
ERROR MESSAGE 10257	FATAL ERROR registered.
ERROR MESSAGE 10258	UNKNOWN ERROR registered.

Error/Warning Messages

The following are possible Error/Warning messages that may be displayed during the operation of MEMDIAG.

MEMORY ERRORs map to one of two different error types: Card Error or Buffer Error. The most important difference is that Card Errors are physical errors and an FRU will be pinpointed; however, Buffer Errors are virtual errors and no FRU can be determined.

```
ERROR MESSAGE 10259

***
***  Unable to retrieve message # ! from MEMDIAG catalog
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE The message number indicated was not found in the message catalog of the diagnostic.

ACTION Get the version number of the diagnostic, indicate what you were attempting to do, and file an SR.

```
ERROR MESSAGE 10260

***
***  Memdiag is unable to OPEN memory, BAD exit status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***  (3) Check to see if /dev/dmem is there, or if /dev/dmem is included in
***      the kernel for an HPUX system.
***
```

CAUSE An operation requested by Memdiag to open memory was not successfully completed due to hardware or software below Memdiag. Usually this means the hardware system is not on (recognized) by the supported list Memdiag officially tests.

ACTION Get the version number of the diagnostic, indicate what you were attempting to do, and file an SR.

WARNING MESSAGE 10261

```
***
***      W A R N I N G :
***
***    It is possible that MEMDIAG will force a parity error to
***    occur.  This may cause the machine to freeze or generate
***    a High Priority Machine Check.
***
```

CAUSE This warning will print out when the memory system has parity data checking instead of EDC error checking. Performing memory tests will exercise memory that may not normally be accessed. This will increase the probability of coming across a memory failure. However, with parity memory checking the error will be detected but not corrected and could, therefore, cause an HPMC.

ACTION This warning is basically for old systems that are running with parity cards (ie: Burgundy), which are not "officially" supported but the warning is printed to inform user of possible results.

WARNING MESSAGE 10262

```
***
***  Bad RANGE input, this range goes over End Of Memory.
***
***       Please enter a SMALLER number.
***
```

CAUSE User has input a Page Range that when added to the current requested Starting Page will put requested pages over valid page numbers in the memory system.

ACTION Input a smaller Page Range value.

ERROR MESSAGE 10263

```
***
***  At the current time, the Operating System was not able to
***  ALLOC the minimum buffer size needed to run MEMDIAG.
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE The minimum buffer required to execute Memdiag was not obtained. the memory space is not available in the present state of the environment. The system is saying it ran out of memory pages to give away.

ACTION Exit the diagnostic and restart again, this will deallocate all memory pages obtained by the diagnostic and try again. If it still does not work collect data about the problem and submit to STD Online Support.

ERROR MESSAGE 10264

```
***
***  At the current time, the Sherlock Interface was not able
***  to obtain a pattern buffer from the Operating System of
***  the minimum buffer size needed to run MEMDIAG.
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE The minimum buffer required to execute Memdiag was not obtained, the memory space is not available in the present state of the environment. The system is saying it ran out of memory pages to grant ownership to.

ACTION Exit the diagnostic and restart again. this will deallocate all memory pages obtained by the diagnostic and try again. If it still does not work collect data about the problem and submit to STD Online Support.

WARNING MESSAGE 10265

```
***
***   At the current time, the Operating System was not
***   able to allocate the buffer size requested so try
***   requesting a smaller ALLOC SIZE parameter.
***
```

CAUSE	At the current time, the operating system does not seem to have the amount of pages necessary to support running Memdiag with the ALLOC SIZE you requested.
ACTION	Input a smaller ALLOC SIZE parameter.

WARNING MESSAGE 10266

```
***
*** WARNING -- You have reached your maximum error total
***
```

CAUSE	The user specified error limit has been reached.
ACTION	If you would like to allow for more errors before termination, re-run the diagnostic assigning a larger value to the errcount parameter of the run command.

WARNING MESSAGE 10267

```
***
*** WARNING -- Invalid response.  Please try again.
***
```

CAUSE — Valid ranges acceptable for this prompt will be displayed either above the prompt line or at the end of the prompt line.

ACTION — Input value from within displayed range.

ERROR MESSAGE 10268

```
***
***  Memdiag has encountered a BUFFER error while testing.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE — This data error maps to Virtual Memory which is outside the scope of the Memory Array Diagnostic. It can indicate a failure anywhere within the HPPA system (ie bus, hard disk, cables, etc).

ACTION — Rerun the test to see if the Buffer Error will repeat itself but this time map to physical memory where we can pinpoint an FRU. If the error does not repeat, chances are you will never know what caused it. If it does repeat, and repeats again as a Buffer Error, run other diagnostics to try and locate the failure within the system.

ERROR MESSAGE 10269

```
***
***  The Driver indicates that there are NO Memory Controllers in the
***  system.  Check that Memory Controller card is properly installed.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE When the driver has gone out to poll the bus line for Memory Controller cards, it has not found one present. This is obviously an impossibility because a MC card must be present within the system for it to function at all; however, the reporting of the MC card can fail in transit to the Memdiag query. This indicates some sort of system failure outside the scope of Memdiag.

ACTION Check for MC card and make sure it is completed pushed into slot, if it still does not work collect data about the problem and submit to STD Online Support.

ERROR MESSAGE 10270

```
***
***  A CARD error was detected and logged in the Syndrome register.
***     The following messages will give details of error location.
***
```

CAUSE A physical memory has occurred during a read or write operation to memory.

ACTION Replace Memory Controller or Memory Array card indentified in the subsequent error messages that will pinpoint the FRU.

ERROR MESSAGE 10271

```
***
***  A BUFFER error was detected during a comparision of Expected
***                      data to Actual Data read back from memory.
***
***     This type of error indicates some system malfunction OTHER than the
***     Memory Array cards or memory on board the Memory Controller card.
***     If these were faulty, the error would have shown during the readback
***     of the Syndrome Error Register.
***
***     The MC and/or MA card slots that map to the ERROR PAGE are given
***     only as an environment "dump" of all configuration information
***     associated with the ERROR PAGE at the time of the BUFFER error.
***
***     It is recommended that you retest the ERROR PAGE or the Page Range
***     of the associated Memory Controller and/or Memory Array where the
***     BUFFER error occurred (go into User Interactive, Section 8).
***
***     The following messages will give details on the BUFFER ERROR.
***
```

CAUSE This data error maps to Virtual Memory which is outside the scope of the Memory Array Diagnostic. It can indicate a failure anywhere within the HPPA system (ie bus, hard disk, cables, etc).

ACTION Rerun the test to see if the Buffer Error will repeat itself but this time map to physical memory where we can pinpoint an FRU. If the error does not repeat, chances are you will never know what caused it. If it does repeat, and repeats again as a Buffer Error, run other diagnostics to try and locate the failure within the system.

```
ERROR MESSAGE 10272

***
***   ERROR found at Page 1840
***
***     Expected Data = 0x        0
***
***     Actual   Data = 0x        1
***
***     Test Pattern  = All Zeros
***
```

CAUSE This data error maps to Virtual Memory which is outside the scope of the Memory Array Diagnostic. It can indicate a failure anywhere within the HPPA system (ie bus, hard disk, cables, etc).

ACTION Rerun the test to see if the Buffer Error will repeat itself but this time map to physical memory where we can pinpoint an FRU. If the error does not repeat, chances are you will never know what caused it. If it does repeat, and repeats again as a Buffer Error, run other diagnostics to try and locate the failure within the system.

```
ERROR MESSAGE 10273

***
***   MEMORY ERROR DETECTED
***
***     Test Pattern                    =   All Zeros
***
***     Bit  Location                   =   63
***     Chip Location                   =   u0301
***     Bank Location                   =   0
***
***     Memory Controller Slot          =   0
***     Memory Array Slot               =   0
***
```

CAUSE A physical memory has occurred during a read or write operation to memory.

ACTION Replace Memory Controller or Memory Array card indentified in the subsequent error messages that will pinpoint the FRU.

ERROR MESSAGE 10275

```
*** Under Memory Controller in Slot 3 maps to the BUFFER ERROR PAGE.
```

CAUSE This data error maps to Virtual Memory which is outside the scope of the Memory Array Diagnostic. It can indicate a failure anywhere within the HPPA system (ie bus, hard disk, cables, etc).

ACTION Rerun the test to see if the Buffer Error will repeat itself but this time map to physical memory where we can pinpoint an FRU. If the error does not repeat, chances are you will never know what caused it. If it does repeat, and repeats again as a Buffer Error, run other diagnostics to try and locate the failure within the system.

ERROR MESSAGE 10276

```
*** Memory Array in Slot A maps to the BUFFER ERROR PAGE.
```

CAUSE This data error maps to Virtual Memory which is outside the scope of the Memory Array Diagnostic. It can indicate a failure anywhere within the HPPA system (ie bus, hard disk, cables, etc).

ACTION Rerun the test to see if the Buffer Error will repeat itself. If the error does not repeat, chances are you will never know what caused it. If it does repeat, and repeats again as a Buffer Error, run other diagnostics to try and locate the failure within the system. If it repeats as a Card Error then the FRU will be identified.

ERROR MESSAGE 10277

```
***
***   FAILURE in INITIALIZATION of Memory Controller
***
***   Replace Memory Controller chip or entire card.
***
```

CAUSE — A physical memory has occurred during a read or write operation to memory.

ACTION — Replace Memory Controller or Memory Array card indentified in the subsequent error messages that will pinpoint the FRU.

ERROR MESSAGE 10278

```
***
***  FAILURE in retrieving Configuration Information
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE — The configuration tables did not print out successfully. This indicates some sort of print utility failure, it will not effect successful completion of diagnostic.

ACTION — Get the version number of the diagnostic, indicate what you were attempting to do, and file an SR.

```
ERROR MESSAGE 10279

***
***   FAILURE in Initialization of Memory
***
```

CAUSE A physical memory has occurred during a read or write operation to memory.
ACTION Replace Memory Controller or Memory Array card indentified in the subsequent error messages that will pinpoint the FRU.

```
ERROR MESSAGE 10280

***
***   FAILURE in Pattern Testing of Memory
***
```

CAUSE A physical memory has occurred during a read or write operation to memory.
ACTION Replace Memory Controller or Memory Array card identified in the subsequent error messages that will pinpoint the FRU.

```
ERROR MESSAGE 10281

***
***    The TROUBLE TREE has NOT completed successfully.
***
```

CAUSE — This indicates an error occurred somewhere in the testing sequence of the Trouble Tree section. This message is the global summation of all test sequences called for Memdiag, in case the User has run overnight and returned to the console to see the last message printed out before Memdiag terminated.

ACTION — Look back through previous messages to find Specific Error Message that will pinpoint a Field Replaceable Unit.

```
ERROR MESSAGE 10282

***
*** Unable to successfully acquire DAR buffer.
*** MEMDIAG will now terminate.
***
*** Action:
*** (1) Try to run the section again with ERRPRINT command modifier. or
*** (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE — An operation requested by Memdiag was not successfully completed due to hardware or software below Memdiag.

ACTION — Get the version number of the diagnostic, indicate what you were attempting to do. and file an SR.

ERROR MESSAGE 10283

```
***
***  Unable to initialize Memdiag in diagostic environment.
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE An operation requested by Memdiag was not successfully completed due to hardware or software below Memdiag.

ACTION Get the version number of the diagnostic, indicate what you were attempting to do, and file an SR.

ERROR MESSAGE 10284

```
***
***  Loading of a Section or Step failed.
***  MEMDIAG will now terminate. Error Status = !
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE An operation requested by Memdiag was not successfully completed due to hardware or software below Memdiag.

ACTION Get the version number of the diagnostic, indicate what you were attempting to do, and file an SR.

WARNING MESSAGE 10285

```
***
***    W A R N I N G
***
***    Selecting GROUPS of patterns in the User Interactive section
***    will effect system performance and increase test duration.
***
***    If you have not already done so, run with ONE pattern to test
***    out your systems  performance before trying multiple pattern tests.
***
```

CAUSE Memory tests are CPU intensive tests that may effect system performance adversely.

ACTION It is suggested you run Memdiag using ONE pattern first to test out system performance under your own unique circumstances of system load, system operating system and specific hardware.

ERROR MESSAGE 10286

```
***
***  Interrupt POLL failed with Status = !
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE An operation requested by Memdiag was not successfully completed due to hardware or software below Memdiag.

ACTION Get the version number of the diagnostic, indicate what you were attempting to do, and file an SR.

WARNING MESSAGE 10287

```
***
***        W A R N I N G
***
***   The section you have selected will run memory tests that
***   take over one half hour to complete (on average).
***
***   If you don't have the time now, go into User Interactive (Sect 8)
***   and setup your own memory test by selecting parameters that will
***   specifically test certain ranges and, therefore, shorten test duration.
***
***   Do you want to continue with this Section now? (y/n) [y]:
***
```

CAUSE — This is to give the CE a way out if he is not aware of the time required to run these memory tests.

ACTION — If you can run the diagnostic in the background, do so. If you are looking for immediate results. go into the User Interactive (Section 8) and run specific test ranges.

ERROR MESSAGE 10289

```
***
***  Memdiag is unable to read Syndrome Register
***  in order to display status,  Bad Exit Status = !
***  MEMDIAG will continue ...
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE — Memdiag requested lower level software to read the Syndrome Register in order to decode the error flags set, if any. The request was not successfully completed. This is only for display so. therefore, Memdiag can continue.

ACTION — Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10290

```
***
***  Memdiag is unable to read Model Information
***  in order to display Revision Info,  Bad Exit Status = !
***  MEMDIAG will continue ...
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE
: Memdiag requested lower level software to read the Model Information in order to display revision information. The request was not successfully completed. This is only for display so, therefore, Memdiag can continue.

ACTION
: Try running the section again. If it still fails. record exit status and submit to STD Online Support.

ERROR MESSAGE 10291

```
***
***  Memdiag is unable to Allocate pages.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE
: Memdiag was not able to successfully allocate a page from the operating system. This means it could not be granted ownership of it in order to test it.

ACTION
: Try running the section again. If it still fails. record exit status and submit to STD Online Support.

ERROR MESSAGE 10292

```
***
***  Memdiag is unable to Deallocate pages.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE — Memdiag was not able to successfully deallocate a page to the operating system. This means it could not be return ownership of it.

ACTION — Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10293

```
***
***  Memdiag is unable to User Allocate pages.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE — Memdiag was not able to successfully allocate a page from the operating system. This means it could not be granted ownership of it in order to test it.

ACTION — Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10294

```
***
***  Memdiag is unable to Clear Syndrome Register.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to clear the Syndrome Register. Memdiag must terminate because if the syndrome can't be cleared it will read an error at each cycle of the testing.

ACTION Try running the section again. If it still fails. record exit status and submit to STD Online Support.

ERROR MESSAGE 10295

```
***
***  Memdiag is unable to read System DETAIL Info.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to read the system detail information in order to test memory. Memdiag cannot test without knowing what memory controller/array cards are present.

ACTION Try running the section again. If it still fails. record exit status and submit to STD Online Support.

ERROR MESSAGE 10296

```
***
***  Memdiag is unable to read System CONFIGURATION Info.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to read the system configuration information in order to test memory. Memdiag cannot test without knowing what memory controller/array cards are present.

ACTION Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10297

```
***
***  Memdiag is unable to read System MODEL Info.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to read the Model Information in order to display revision information. The request was not successfully completed. This is needed for Memdiag to order the SPAs of each memory range, therefore, Memdiag cannot continue without it.

ACTION Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10298

```
***
***  Memdiag is unable to read Syndrome Register.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to read the Syndrome Register in order to decode the error flags set, if any. The request was not successfully completed. This is for error decoding after the test and is required.

ACTION Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10299

```
***
***  Memdiag is unable to READ Memory.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to read the memory after a pattern test write.

ACTION Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10300

```
***
***  Memdiag is unable to WRITE Memory.  Bad Exit Status = !
***  MEMDIAG will now terminate.
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to write the memory for pattern testing.

ACTION Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10301

```
***
***  Memdiag is unable to read Page Status for page !
***  Bad Exit Status = !,  MEMDIAG will continue ...
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE Memdiag requested lower level software to read the type of page status associated with the indicated page.

ACTION Try running the section again. If it still fails, record exit status and submit to STD Online Support.

```
ERROR MESSAGE 10304

***
***     Memdiag is unable to log memory error to MEMLOGP.
***     Bad Exit Status = !,  MEMDIAG will continue ...
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE	Memdiag sent an IPC message to log a memory error to MEMLOG file, but MEMLOGP was not running.
ACTION	Type in DIAGSYSTEM RUN MEMLOGP at DUI prompt. Try running the section again. If it still fails, record exit status and submit to STD Online Support.

```
ERROR MESSAGE 10306

***  Bank Number 0 maps to the BUFFER ERROR PAGE.
```

CAUSE	This data error maps to Virtual Memory which is outside the scope of the Memory Array Diagnostic. It can indicate a failure anywhere within the HPPA system (ie bus, hard disk, cables, etc).
ACTION	Rerun the test to see if the Buffer Error will repeat itself. If the error does not repeat, chances are you will never know what caused it. If it does repeat, and repeats again as a Buffer Error, run other diagnostics to try and locate the failure within the system. If it repeats as a Card Error then the FRU will be identified.

ERROR MESSAGE 10309

```
***
***  Memdiag is unable to get Page Size from MEMDAR
***  Bad Exit Status = !,  MEMDIAG will not continue ...
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE	Memdiag requested lower level software to obtain page size for page allocation.
ACTION	Try running the section again. If it still fails, record exit status and submit to STD Online Support.

ERROR MESSAGE 10310

```
***
***  Memdiag is unable to get Memory Subsystem Type from MEMDAR
***  Bad Exit Status = !,  MEMDIAG will not continue ...
***
***  Action:
***  (1) Try to run the section again with ERRPRINT command modifier. or
***  (2) Try to run the section again with TRACE=ALL command modifier.
***
```

CAUSE	Memdiag requested lower level software to obtain memory subsystem type for deciding which subsystem test path to take.
ACTION	Try running the section again. If it still fails, record exit status and submit to STD Online Support.

5

A1020A 2D Graphics Subsystem Diagnostic

Introduction

The A1020A 2D Graphics Subsystem Diagnostic (`GS2DDIAG`) provides an online test of A1020A Graphics Subsystem on any HP Precision Architecture RISC computer system which supports the Online Diagnostic subsystem. The A1020A must be connected to a A1017A card. The A1020A along with a corresponding high resolution color graphics terminal can be configured as the system console or a graphics workstation.

When a test function associated with a particular board has failed, replacement of that Field Replaceable Unit (FRU) is indicated.

Defects and Enhancements

Submit defect reports and enhancement requests for this diagnostic through the STARS database referencing product number 30600-10016.

Minimum Configuration

`GS2DDIAG` is designed to test and verify any A1020As configured as system consoles or workstations on any PA-RISC computer. The system must be up and running HP-UX and have a A1017A board correctly installed and configured to the A1020A.

Operating Instructions

`GS2DDIAG` is accessed by the user via the Diagnostic User Interface.

Default Tests

If the user does not specify which sections to run, the default is to run all of the test sections.

RUN Command

To bring up the Online Diagnostic subsystem, enter the following command to the HP-UX system prompt:

`%sysdiag`

The system responds with the following prompt indicating that access has been gained to the Diagnostic User Interface (DUI).

`DUI >`

Typing **HELP** causes a summary of the **DUI** and its commands to be printed. Refer to the **DUI** chapter of this manual for details.

Note

The device to be tested must be powered up and on line. Device physical locations (pdev) shown in the **run** commands are those of the devices on the "typical A1002A" system configuration described in the chapter on **DUI**. The pdev value entered must be correct for the system being tested.

For example, to run the diagnostic, you might enter:

```
DUI >run gs2ddiag pdev=8 <RUN Command Options>
                        |           |
                        |     none required for
                        |     default test suite
                        |
              insert physical location of
              device to be tested here,
              or type the devfile name
```

The **devfile** command must use one of the following values to identify the midbus slot where the A1017A resides:

`crt0` - Midbus slot 2
`crt1` - Midbus slot 3
`crt2` - Midbus slot 4
`crt3` - Midbus slot 5

Test Execution

When GS2DDIAG is run, a header and a welcome message will be displayed.

Note If the user runs GS2DDIAG from the graphics subsystem as the system console, CTRL -C should be pressed to interrupt the diagnostic; the CTRL SHIFT RESET keys should also be pressed to reset the graphics subsystem, and to allow GS2DDIAG to return to the DUI> prompt.

Because all sections are either disruptive or destructive, the diagnostic requires that all sections be run in Single User Mode (SUM). Only those users with level 0 security will be able to execute all sections.

If at any time, the number of errors generated reaches the limit specified by the user (10 for instance) in the ERRCOUNT parameter of the **run** command, the following message will be output:

```
Error count exceeded. Count = 10
```

The diagnostic will then terminate execution. If the TRACE parameter is used and equated to PROGRAMS, extended informational messages are displayed along with the diagnostic output. If the diagnostic is run from the HP98556A which is being tested, the TRACE info should be sent to a file to avoid corrupting visual output.

If the ERRPAUSE parameter of the **run** command was assigned a value of "on", then this diagnostic will stop after each error is generated and ask the user if testing should continue:

```
Error pause -- do you wish to continue ([y]/n) ?
```

If the response is "Y", then the test will be resumed (if possible), if the response is "N", then the diagnostic terminates. The "[Y]" indicates that "Y" is the default response if the user simply hits Return in response to the prompt. If the sections specified by the user were executed the number of times specified in the LOOP parameter of the **run** command without the number of errors exceeding the ERRNUM value, the diagnostic terminates normally. Upon termination of this diagnostic, control returns to the diagnostic system.

Test Section Descriptions

GS2DDIAG has the following sections available:

Section 10	Cycle Type Register Test
Section 11	Address Register Test
Section 13	98550A ID ROM Checksum Test
Section 14	Word Mode Access Test
Section 15	Byte Mode Access Test
Section 16	Long Word Mode Access Test
Section 22	Register R/W Test
Section 23	Color Map Initialization
Section 24	Frame Buffer Read/Write Test
Section 25	BARC Chip(s) Test
Section 26	RUG Chip Test
Section 27	Final Pattern Generation and IRIS Color Map Read/Write Test
Section 30	FASTCAT SPU Memory Tests
Section 31	FASTCAT GPU Memory Tests
Section 32	FASTCAT SPU/GPU DRAM Memory Tests
Section 33	FASTCAT SPU/GPU DRAM Simultaneous Memory Tests
Section 34	FASTCAT SPU/GPU DRAM/SRAM Simultaneous Memory Tests
Section 35	FASTCAT SPU/GPU SRAM Memory Tests
Section 36	FASTCAT SPU/GPU SRAM Simultaneous Memory Tests
Section 37	FASTCAT Transform Engine Tests
Section 38	FASTCAT Graphics Processor Test

Each section includes an explanation which consists of the actions performed and the expected output if no errors occur. For a complete list of error and information messages that may be encountered while running GS2DDIAG, refer to the Information/Error section at the end of this chapter. The "!" in the Output displays indicates that a parameter of some sort will replace the exclamation point when the message is displayed.

Section 10—Cycle Type Register Test

This section verifies the register that determines whether the A1020A will generate byte, word, or long word cycles.

Output:

```
CYCLE TYPE REGISTER TEST FAILED.
```

Section 11—Address Register Test

This section verifies the A1020A block address register functionality.

Output:

```
BLOCK ADDRESS REGISTER TEST FAILED.
```

Section 13—98550A ID ROM Checksum Test

This section verifies that the the A1020A interface can communicate with the system's ID ROM.

Output:

```
ID ROM TEST FAILED; BOARD .
```

Section 14—Word Mode Access Test

This section verifies the A1020A word mode access functionality in more detail.

Output:

```
LS BYTE WROTE !, LS BYTE READ !; BOARD !
LS BYTE WROTE !, MS BYTE READ !; BOARD !
MS BYTE WROTE !, MS BYTE READ !; BOARD !
MS BYTE WROTE !, LS BYTE READ !; BOARD !
FIREYE WORD MODE TEST FAILED; BOARD !.
```

Section 15—Byte Mode Access Test

This section verifies the A1020A byte mode access functionality.

Output:

```
LOWER BYTE WROTE !, LOWER BYTE READ !; BOARD !
LOWER BYTE WROTE !, UPPER BYTE READ !; BOARD !
UPPER BYTE WROTE !, UPPER BYTE READ !; BOARD !
UPPER BYTE WROTE !, LOWER BYTE READ !; BOARD !
FIREYE BYTE MODE TEST FAILED; BOARD !.
```

Section 16—Long Word Mode Access Test

This section verifies the A1020A long word mode access functionality and checks the integrity of the 16 most significant bits of the data bus.

Output:

```
FIREYE LONG MODE READ 0x!, INSTEAD OF 0x! AT 0x!.
LONG MODE TEST FAILED; BOARD !.
```

Section 22—Register R/W Test

This section executes a read/write test on all system hardware registers including:

- Read only registers
- Standard read/write registers
- BARC pattern RAM
- BARC TCREN registers
- BARC PRR, WRR and TRR registers

Output:

```
ON REG Ox! EXPECTED Ox! AND GOT Ox!.
REGISTER R/W TEST FAILED.
```

Section 23—Color Map Initialization

This section sets up the IRIS color map chip so that all following tests may be observed on the monitor.

Output:

```
SHORT IRIS TEST FAILED COLORMAP INITIALIZATION.
SHORT IRIS TEST FAILED OVERLAY INITIALIZATION.
SHORT IRIS TEST FAILED COLORMAP INITIALIZATION RECHECK.
SHORT IRIS TEST FAILED OVERLAY INITIALIZATION RECHECK.
SHORT IRIS TEST FAILED.
```

Section 24—Frame Buffer Read/Write Test

This section verifies that every pixel location can be written and read with a 1 or 0 without affecting adjacent pixels. The order of events is:

- Clear frame buffer and verify frame buffer cleared.
- Walking 1 test.
- Set frame buffer and verify frame buffer set.
- Walking 0 test.

Output:

```
AT Ox! EXPECTED Ox! AND GOT Ox!.
BUS ERROR DOING WORD TRANSFERS TO THE FRAME BUFFER.
BUS ERROR DOING BYTE TRANSFERS TO THE FRAME BUFFER.
FRAME BUFFER TEST FAILED.
```

Section 25—BARC Chip(s) Test

This section verifies BARC chip functionality. The number of patterns for each display is:

- HRC : 10 displayed pattern, 1 undisplayed pattern.
- LCC : 6 displayed patterns.
- MONO: 2 displayed patterns.

Functions tested:

- ALU operations
- Bit/Byte per pixel accesses
- Pattern RAM operations
- Write and Read-Modify-Write cycles

Output:

```
IRIS SETUP FAILED IN BARC TEST.
CLEAR SCREEN FAILED IN BARC TEST.
BARC ! FAILED.
BARC FAILED INITIALIZATION.
BARC FAILED SPU BYTE/PIXEL WRITES.
BARC FAILED SPU BIT/PIXEL WRITES.
BARC FAILED PATTERN RAM AND TRR TEST.
BARC ! FAILED CRC CHECK.
BARC TEST FAILED.
```

Section 26—RUG Chip Test

This section verifies RUG chip functionality including:

- Vector drawing (includes linetype and non-linetype vectors)
- Circle drawing (includes linetype, non-linetype and filled circles)
- Block moves
- Polygon area fill
- Picking (includes primitive clipping)

Output:

```
RUG SCREEN CLEAR FAILED.
RUG VECTOR TEST FAILED.
RUG VECTOR TEST FAILED CRC CHECK.
RUG CIRCLE TEST FAILED.
RUG CIRCLE TEST FAILED CRC CHECK.
RUG BLIT TEST FAILED.
RUG BLIT TEST FAILED CRC CHECK.
RUG FILL TEST FAILED.
RUG FILL TEST FAILED CRC CHECK.
RUG PICK TEST FAILED.
RUG PICK TEST FAILED CRC CHECK.
RUG CRC CHECK FAILED.
RUG READ 0x! INSTEAD OF 0x! AT 0x!.
RUG TEST FAILED.
```

Section 27—Final Pattern Generation and IRIS Color Map Read/Write Test

This section generates a final pattern which can be used to verify monitor convergence.

Output:

```
FINAL PATTERN TEST SCREEN CLEAR FAILED.
FINAL PATTERN TEST PATTERN CREATION FAILED.
FINAL PATTERN CRC FAILED ON MAIN PLANES.
FINAL PATTERN CRC FAILED ON OVERLAY PLANES.
FINAL PATTERN GENERATION FAILED.
IRIS TEST FAILED COLORMAP R/W TEST.
IRIS TEST FAILED OVERLAY R/W TEST.
IRIS TEST FAILED.
```

Section 30 : FASTCAT SPU Memory Tests

In this section the system processor (SPU) is used to test all the FASTCAT RAM. The section includes separate tests for the DRAM, which is used for downloaded graphics processor code, and the SRAM, which is used to pass data and commands between the two processors. This test covers all of both RAM areas. Other sections perform more extensive SPU tests on parts of the RAM and complete GPU RAM tests. This section includes tests 1 and 2. Run time is 4:09 seconds.

Output:

```
FASTCAT SPU MEMORY TEST FAILED; BOARD !.
```

Section 31 : FASTCAT GPU Memory Tests

In this section the GPU is used to perform complete tests on both the DRAM and SRAM in FASTCAT. GPU firmware is loaded into the DRAM to run the tests. Since the GPU uses the DRAM for its programs and the SRAM for communication with the SPU, each block of RAM is divided in half and tested separately. RAM failures may appear either as memory test errors or as GPU timeouts. This section includes tests 3, 4, 5, and 6. Run time is 55 second.

Output:

```
FASTCAT GPU MEMORY TEST FAILED; BOARD !.
GPU MEMORY FAILURE: TEST #! - ADDRESS 0x! EXPECTED 0x! ACTUAL 0x!.
```

Section 32 : FASTCAT SPU/GPU DRAM Memory Tests

In this section the GPU and the SPU are used to perform sequential tests on the FASTCAT DRAM. First the SPU writes to the DRAM, then the GPU reads it back and checks it. Then the GPU writes the RAM, and the SPU checks it. This section includes tests 7 and 8. Run time is 31 second.

Output:

```
FASTCAT COMBINED SPU/GPU TEST OF DRAM FAILED; BOARD !.
```

Section 33 : FASTCAT SPU/GPU DRAM Simultaneous Memory Tests

In this section the GPU and the SPU are used to perform simultaneous tests on the FASTCAT DRAM. Since the GPU program must reside in the DRAM, the memory is divided in half. First the SPU tests the lower half of the RAM while the GPU tests the upper half, then they switch. This section is not part of the default test. This section includes tests 9 and 10. Run time is 5:11.

Output:

```
FASTCAT SIMULTANEOUS SPU/GPU TEST OF DRAM FAILED; BOARD !.
```

Section 34 : FASTCAT SPU/GPU DRAM/SRAM Simultaneous Memory Tests

In this section the GPU and the SPU are used to perform simultaneous tests on the FASTCAT DRAM and SRAM. First the SPU tests the DRAM while the GPU tests the SRAM, then they switch. This section is not part of the default test. This section includes tests 11 and 16. Run time is 3:19.

Output:

```
FASTCAT SIMULTANEOUS SPU/GPU TEST OF DRAM AND SRAM FAILED; BOARD !.
```

Section 35 : FASTCAT SPU/GPU SRAM Memory Tests

In this section the GPU and the SPU are used to perform sequential tests on the FASTCAT SRAM. One processor writes the RAM, then the other reads it back. The actual tests are the same as in Section 32. This section includes tests 12 and 13. Run time is 1 second.

Output:

```
FASTCAT COMBINED SPU/GPU TEST OF SRAM FAILED; BOARD !.
```

Section 36 : FASTCAT SPU/GPU SRAM Simultaneous Memory Tests

In this section the GPU and the SPU are used to perform simultaneous tests on the FASTCAT SRAM. One processor tests the lower half of the RAM while the other tests the upper half, then they switch. The actual tests are the same as in Section 33. This section includes tests 14 and 15. Run time is 1:12.

Output:

```
FASTCAT SIMULTANEOUS SPU/GPU TEST OF SRAM FAILED; BOARD !.
```

Section 37 : FASTCAT Transform Engine Tests

This section tests the integer Transform Engine in FASTCAT. Run time is 26 seconds. The functions tested include:

- Revision - verify valid microcode ROM revision number
- Nontrivial - non-orthogonal vector clipping
- Forytfive - forty five degree vectors
- Trivial - horizontal and vertical vectors
- Rotate - vector rotation
- Pline overflow - check for out of range polyline endpoints
- Transform point - point transform operations
- Superdc - direct path to RUG vector generator
- DC pline - device coordinate polylines
- Transform pline - world coordinate (transformed) polylines
- Register - math chip registers
- Trivial pline - horizontal and vertical polylines
- Transform rectangle - world coordinate (transformed) rectangles
- DC rectangle - device coordinate rectangles
- Translate rectangle - window translation of rectangles
- Small polygon - polygons with 1 and 2 pixel edge lengths
- Polygon overflow - check for out of range polygon vertices
- DC polygon - device coordinate polygons
- Polygon translate - window translation of polygons
- HV polygon - polygons with horizontal and vertical sides
- Polygon rotate - polygon rotation
- Circles - circle operations

Output:

```
FASTCAT TRANSFORM ENGINE TEST FAILED; BOARD !.
```

Section 38 : FASTCAT Graphics Processor Test

This section tests the GPU's ability to run a program, process a command list, and draw to the screen. The test downloads firmware to the GPU, and then starts the GPU. The GPU generates and draws two images, then the SPU reads and CRCs the frame buffer. Run time is 44 seconds.

Output:

```
GPU TEST NO. ! FAILED. CRC SHOULD BE !; WAS !. CHECKSUM SHOULD BE !; WAS !.
GPU ERROR; LEVEL = !; ERROR NO. = !.
FASTCAT GPU TEST FAILED; BOARD !.
```

Error Messages

The following error messages may appear when using GS2DDIAG. The "!" indicates that a parameter of some sort will replace the exclamation point when the error message is displayed.

2000 `Caught termination signal. (GS2DMSG 2000)`
CAUSE
ACTION

2006 `Test_index ! bank ! passed !. (GS2DMSG 2006)`
CAUSE
ACTION

2007 `Attempting to map card !. (GS2DMSG 2007)`
CAUSE
ACTION

2008 `Attempting to unmap card. (GS2DMSG 2008)`
CAUSE
ACTION

2009 `Writing data to stdout. (GS2DMSG 2009)`
CAUSE
ACTION

2010 `! !`
CAUSE
ACTION

2011 `Board type is HRC. (GS2DMSG 2011)`
CAUSE
ACTION

2012 `Board type is LCC. (GS2DMSG 2012)`
CAUSE
ACTION

2013	Board type is MONOCHROME. (GS2DMSG 2013)
CAUSE	
ACTION	

2014	Board type is UNKNOWN. (GS2DMSG 2014)
CAUSE	
ACTION	

2015	CRC CHECKSUM (GS2DMSG 2015)
CAUSE	
ACTION	

2016	98550A #! present at !. (GS2DMSG 2016)
CAUSE	
ACTION	

2017	Attempting to test card !. (GS2DMSG 2017)
CAUSE	
ACTION	

2018	Checking the A1020A cycle type register. (GS2DMSG 2018)
CAUSE	
ACTION	

2019	Checking the A1020A block address register. (GS2DMSG 2019)
CAUSE	
ACTION	

2020	Looking for board !. (GS2DMSG 2020)
CAUSE	
ACTION	

2021 Checking the A1020A word mode accesses to board !. (GS2DMSG 2021)
CAUSE
ACTION

2022 Checking the A1020A byte mode accesses to board !. (GS2DMSG 2022)
CAUSE
ACTION

2023 Checking board type. (GS2DMSG 2023)
CAUSE
ACTION

2024 Executing display ID ROM. (GS2DMSG 2024)
CAUSE
ACTION

2025 Checking display ID ROM on board !. (GS2DMSG 2025)
CAUSE
ACTION

2026 Checking chip revisions. (GS2DMSG 2026)
CAUSE
ACTION

2027 Checking sync registers. (GS2DMSG 2027)
CAUSE
ACTION

2028 Checking read-only registers. (GS2DMSG 2028)
CAUSE
ACTION

2029 `Checking standard registers. (GS2DMSG 2029)`
CAUSE
ACTION

2030 `Checking barc registers that vary with board type. (GS2DMSG 2030)`
CAUSE
ACTION

2031 `Checking pattern registers. (GS2DMSG 2031)`
CAUSE
ACTION

2032 `Checking TCREN registers. (GS2DMSG 2032)`
CAUSE
ACTION

2033 `Checking xRR0 registers. (GS2DMSG 2033)`
CAUSE
ACTION

2034 `Checking xRR1 registers. (GS2DMSG 2034)`
CAUSE
ACTION

2035 `Checking main framebuffer. (GS2DMSG 2035)`
CAUSE
ACTION

2036 `Clearing main framebuffer. (GS2DMSG 2036)`
CAUSE
ACTION

2037	`Walking ones thru main framebuffer. (GS2DMSG 2037)`
CAUSE	
ACTION	

2038	`Setting main framebuffer. (GS2DMSG 2038)`
CAUSE	
ACTION	

2039	`Walking zeros thru main framebuffer. (GS2DMSG 2039)`
CAUSE	
ACTION	

2040	`Checking overlay framebuffer. (GS2DMSG 2040)`
CAUSE	
ACTION	

2041	`Walking ones thru overlay framebuffer. (GS2DMSG 2041)`
CAUSE	
ACTION	

2042	`Setting overlay framebuffer. (GS2DMSG 2042)`
CAUSE	
ACTION	

2043	`Walking zeros thru overlay framebuffer. (GS2DMSG 2043)`
CAUSE	
ACTION	

2044	`Checking LCC phantom plane. (GS2DMSG 2044)`
CAUSE	
ACTION	

2045 `Walking ones thru phantom plane. (GS2DMSG 2045)`
CAUSE
ACTION

2046 `Setting phantom plane. (GS2DMSG 2046)`
CAUSE
ACTION

2047 `Walking zeros thru phantom plane. (GS2DMSG 2047)`
CAUSE
ACTION

2048 `Starting BARC test. (GS2DMSG 2048)`
CAUSE
ACTION

2051 `Vbarc_num ! bank_num !. (GS2DMSG 2051)`
CAUSE
ACTION

2052 `Building test set data structure. (GS2DMSG 2052)`
CAUSE
ACTION

2053 `Testing sub module !. (GS2DMSG 2053)`
CAUSE
ACTION

2054 `Generating final pattern. (GS2DMSG 2054)`
CAUSE
ACTION

2055 Initializing IRIS colormap RAM. (GS2DMSG 2055)
CAUSE
ACTION

2056 Initializing IRIS overlay RAM. (GS2DMSG 2056)
CAUSE
ACTION

2057 Rechecking IRIS colormap RAM initialization. (GS2DMSG 2057)
CAUSE
ACTION

2058 Rechecking IRIS overlay RAM initialization. (GS2DMSG 2058)
CAUSE
ACTION

2059 Checking IRIS colormap RAM. (GS2DMSG 2059)
CAUSE
ACTION

2060 Checking IRIS overlay RAM. (GS2DMSG 2060)
CAUSE
ACTION

5000 the A1020A INTERFACE BOARD FAILED. (GS2DERR 5000)
CAUSE
ACTION

5001 UNABLE TO ADDRESS 98550A ! (GS2DERR 5001)
CAUSE
ACTION

5002 98550A DISPLAY BOARD ! FAILED. (GS2DERR 5002)
CAUSE
ACTION

5003 FRAME BUFFER CRC FAILED FIRST ATTEMPT. (GS2DERR 5003)
CAUSE
ACTION

5004 BUS ERROR DURING FRAME BUFFER READ AT ! (GS2DERR 5004)
CAUSE
ACTION

5005 AT FB 0x! EXPECTED 0x! AND GOT 0x! (GS2DERR 5005)
CAUSE
ACTION

5006 BUS ERROR DOING WORD TRANSFERS TO THE FRAME BUFFER (GS2DERR 5006)
CAUSE
ACTION

5007 BUS ERROR DURING the A1020A REGISTER READ AT 0x! (GS2DERR 5007)
CAUSE
ACTION

5008 BUS ERROR DURING REGISTER READ AT 0x! (GS2DERR 5008)
CAUSE
ACTION

5009 BUS ERROR DURING REGISTER BYTE READ AT 0x! (GS2DERR 5009)
CAUSE
ACTION

5010 BUS ERROR DURING FB WORD READ AT 0x! (GS2DERR 5010)
CAUSE
ACTION

5011 BUS ERROR DURING FB BYTE READ AT 0x! (GS2DERR 5011)
CAUSE
ACTION

5012 BUS ERROR DURING the A1020A REGISTER WRITE AT 0x! (GS2DERR 5012)
CAUSE
ACTION

5013 BUS ERROR DURING REGISTER WRITE AT 0x! (GS2DERR 5013)
CAUSE
ACTION

5014 BUS ERROR DURING REGISTER BYTE WRITE AT 0x! (GS2DERR 5014)
CAUSE
ACTION

5015 BUS ERROR DURING FB WORD WRITE AT 0x! (GS2DERR 5015)
CAUSE
ACTION

5016 BUS ERROR DURING FB BYTE WRITE AT 0x! (GS2DERR 5016)
CAUSE
ACTION

5017 ON REG 0x! EXPECTED 0x! AND GOT 0x! (GS2DERR 5017)
CAUSE
ACTION

5018 CYCLE TYPE REGISTER TEST FAILED. (GS2DERR 5018)
CAUSE
ACTION

5019 BLOCK ADDRESS REGISTER TEST FAILED. (GS2DERR 5019)
CAUSE
ACTION

5020 NO 98550A FOUND. (GS2DERR 5020)
CAUSE
ACTION

5021 ID ROM TEST FAILED; BOARD ! (GS2DERR 5021)
CAUSE
ACTION

5022 WORD MODE TEST FAILED; BOARD ! (GS2DERR 5022)
CAUSE
ACTION

5023 BYTE MODE TEST FAILED; BOARD ! (GS2DERR 5023)
CAUSE
ACTION

5024 98550A BOARD CANNOT BE INITIALIZED."); (GS2DERR 5024)
CAUSE
ACTION

5025 REGISTER R/W TEST FAILED. (GS2DERR 5025)
CAUSE
ACTION

5026 SHORT IRIS TEST FAILED. (GS2DERR 5026)
CAUSE
ACTION

5027 FRAME BUFFER TEST FAILED. (GS2DERR 5027)
CAUSE
ACTION

5028 BARC TEST FAILED. (GS2DERR 5028)
CAUSE
ACTION

5029 RUG TEST FAILED. (GS2DERR 5029)
CAUSE
ACTION

5030 FINAL PATTERN GENERATION FAILED. (GS2DERR 5030)
CAUSE
ACTION

5031 IRIS TEST FAILED. (GS2DERR 5031)
CAUSE
ACTION

5032 LS BYTE WROTE !, LS BYTE READ !; BOARD ! (GS2DERR 5032)
CAUSE
ACTION

5033 LS BYTE WROTE !, MS BYTE READ !; BOARD ! (GS2DERR 5033)
CAUSE
ACTION

5034 MS BYTE WROTE !, MS BYTE READ !; BOARD ! (GS2DERR 5034)
CAUSE
ACTION

5035 MS BYTE WROTE !, LS BYTE READ !; BOARD ! (GS2DERR 5035)
CAUSE
ACTION

5036 LOWER BYTE WROTE !, LOWER BYTE READ !; BOARD ! (GS2DERR 5036)
CAUSE
ACTION

5037 LOWER BYTE WROTE !, UPPER BYTE READ !; BOARD ! (GS2DERR 5037)
CAUSE
ACTION

5038 UPPER BYTE WROTE !, UPPER BYTE READ !; BOARD ! (GS2DERR 5038)
CAUSE
ACTION

5039 UPPER BYTE WROTE !, LOWER BYTE READ !; BOARD ! (GS2DERR 5039)
CAUSE
ACTION

5040 ID ROM CHECKSUM ERROR ON FIRST 8K BYTES: BOARD ! (GS2DERR 5040)
CAUSE
ACTION

5041 ID ROM CHECKSUM ERROR ON ENTIRE ROM; BOARD ! (GS2DERR 5041)
CAUSE
ACTION

5044 BUS ERROR DOING BYTE TRANSFERS TO THE FRAME BUFFER (GS2DERR 5044)
CAUSE
ACTION

5045 IRIS SETUP FAILED IN BARC TEST. (GS2DERR 5045)
CAUSE
ACTION

5046 CLEAR SCREEN FAILED IN BARC TEST. (GS2DERR 5046)
CAUSE
ACTION

5047 BARC ! FAILED. (GS2DERR 5047)
CAUSE
ACTION

5048 BARC ! FAILED CRC CHECK. (GS2DERR 5048)
CAUSE
ACTION

5049 BARC FAILED INITIALIZATION. (GS2DERR 5049)
CAUSE
ACTION

5050 BARC FAILED SPU BYTE/PIXEL WRITES. (GS2DERR 5050)
CAUSE
ACTION

5051 BARC FAILED SPU BIT/PIXEL WRITES. (GS2DERR 5051)
CAUSE
ACTION

5052 BARC FAILED PATTERN RAM AND TRR TEST. (GS2DERR 5052)
CAUSE
ACTION

5053 BARC FAILED SPU BYTE/PIXEL WRITES. (GS2DERR 5053)
CAUSE
ACTION

5054 RUG SCREEN CLEAR FAILED. (GS2DERR 5054)
CAUSE
ACTION

5055 RUG CRC CHECK FAILED. (GS2DERR 5055)
CAUSE
ACTION

5056 RUG VECTOR TEST FAILED. (GS2DERR 5056)
CAUSE
ACTION

5057 RUG VECTOR TEST FAILED CRC CHECK. (GS2DERR 5057)
CAUSE
ACTION

5058 RUG CIRCLE TEST FAILED. (GS2DERR 5058)
CAUSE
ACTION

5059 RUG CIRCLE TEST FAILED CRC CHECK. (GS2DERR 5059)
CAUSE
ACTION

5060 RUG BLIT TEST FAILED. (GS2DERR 5060)
CAUSE
ACTION

5061 RUG BLIT TEST FAILED CRC CHECK. (GS2DERR 5061)
CAUSE
ACTION

5062 RUG FILL TEST FAILED. (GS2DERR 5062)
CAUSE
ACTION

5063 RUG FILL TEST FAILED CRC CHECK. (GS2DERR 5063)
CAUSE
ACTION

5064 RUG PICK TEST FAILED. (GS2DERR 5064)
CAUSE
ACTION

5065 RUG PICK TEST FAILED CRC CHECK. (GS2DERR 5065)
CAUSE
ACTION

5066 RUG READ 0x! INSTEAD OF 0x! AT 0x!. (GS2DERR 5066)
CAUSE
ACTION

5067 FINAL PATTERN TEST SCREEN CLEAR FAILED. (GS2DERR 5067)
CAUSE
ACTION

5068 FINAL PATTERN CRC FAILED ON MAIN PLANES. (GS2DERR 5068)
CAUSE
ACTION

5069 FINAL PATTERN CRC FAILED ON OVERLAY PLANES. (GS2DERR 5069)
CAUSE
ACTION

5070 FINAL PATTERN TEST PATTERN CREATION FAILED. (GS2DERR 5070)
CAUSE
ACTION

5071 SHORT IRIS TEST FAILED COLORMAP INITIALIZATION. (GS2DERR 5071)
CAUSE
ACTION

5072 SHORT IRIS TEST FAILED OVERLAY INITIALIZATION. (GS2DERR 5072)
CAUSE
ACTION

5073 SHORT IRIS TEST FAILED COLORMAP INITIALIZATION RECHECK. (GS2DERR 5073)
CAUSE
ACTION

5074 SHORT IRIS TEST FAILED OVERLAY INITIALIZATION RECHECK. (GS2DERR 5074)
CAUSE
ACTION

5075 IRIS TEST FAILED COLORMAP R/W TEST. (GS2DERR 5075)
CAUSE
ACTION

5076 IRIS TEST FAILED OVERLAY R/W TEST. (GS2DERR 5076)
CAUSE
ACTION

5100 ILLEGAL ERROR REPORTING LEVEL. (GS2DERR 5100)
CAUSE
ACTION

5101 CANNOT MAP BOARD: !. (GS2DERR 5101)
CAUSE
ACTION

5102 BUS_ERROR: CAUGHT BUS ERROR. (GS2DERR 5102)
CAUSE
ACTION

5103 CANNOT OPEN CONFIGURATION FILE !. (GS2DERR 5103)
CAUSE
ACTION

5104 CANNOT OPEN ERROR LOGGING FILE !. (GS2DERR 5104)
CAUSE
ACTION

5105 CRC_CHECKSUM: BARC INITIALIZATION FAILED. (GS2DERR 5105)
CAUSE
ACTION

5106 CRC_CHECKSUM: FRAME BUFFER READ FAILED. (GS2DERR 5106)
CAUSE
ACTION

5107	FBCHECK: FIRST FBCHECK FAILED. (GS2DERR 5107)
CAUSE	
ACTION	
5108	FBCHECK: SECOND FBCHECK FAILED. (GS2DERR 5108)
CAUSE	
ACTION	
5109	MAP_CARD: UNABLE TO OPEN !. (GS2DERR 5109)
CAUSE	
ACTION	
5110	MAP_CARD: CHECK OF DEVICE FAILED. (GS2DERR 5110)
CAUSE	
ACTION	
5111	MAP_CARD: MAP OF DEVICE FAILED. (GS2DERR 5111)
CAUSE	
ACTION	
5112	READ_SCAN_LINE: BUS ERROR OCCURRED AT WORD OX!. (GS2DERR 5112)
CAUSE	
ACTION	
5118	WRITE_the A1020A: BUS ERROR DURING WRITE OF OX!. (GS2DERR 5118)
CAUSE	
ACTION	
5130	WRITE_LINE: FB BUS ERROR AT OX!. (GS2DERR 5130)
CAUSE	
ACTION	

5135	RUN_COMMAND_STREAM: BUS ERROR. (GS2DERR 5135)
CAUSE	
ACTION	

5200	FASTCAT ACCELERATOR BOARD FAILED. (GS2DERR 5200)
CAUSE	
ACTION	

5201	FIREYE LONG MODE READ 0x!, INSTEAD OF 0x! AT 0x!. (GS2DERR 5201)
CAUSE	
ACTION	

5202	LONG MODE TEST FAILED; BOARD !. (GS2DERR 5202)
CAUSE	
ACTION	

5203	BUS ERROR DURING LONG WORD READ AT 0x!. (GS2DERR 5203)
CAUSE	
ACTION	

5204	BUS ERROR DURING LONG WORD WRITE AT 0x!. (GS2DERR 5204)
CAUSE	
ACTION	

5205	WAIT FOR FASTCAT TIMED OUT. (GS2DERR 5205)
CAUSE	
ACTION	

5206	WAIT FOR PARSER TIMED OUT. (GS2DERR 5206)
CAUSE	
ACTION	

5207	WAIT FOR IRIS TIMED OUT. (GS2DERR 5207)
CAUSE	
ACTION	

5208	WAIT FOR RUG TIMED OUT. (GS2DERR 5208)
CAUSE	
ACTION	

5209	WAIT FOR COMMAND BUFFER FLUSH TIMED OUT. (GS2DERR 5209)
CAUSE	
ACTION	

5210	WAIT FOR GPU TIMED OUT. (GS2DERR 5210)
CAUSE	
ACTION	

5211	WAIT FOR FATE OUTPUT BUFFER FULL TIMED OUT. (GS2DERR 5211)
CAUSE	
ACTION	

5212	FASTCAT SPU MEMORY TEST FAILED; BOARD !. (GS2DERR 5212)
CAUSE	
ACTION	

5213	FASTCAT GPU MEMORY TEST FAILED; BOARD !. (GS2DERR 5213)
CAUSE	
ACTION	

5214	GPU MEMORY FAILURE: TEST #! - ADDRESS 0x! EXPECTED 0x! ACTUAL 0x!. (GS2DERR 5214)
CAUSE	
ACTION	

5215 FASTCAT COMBINED SPU/GPU TEST OF DRAM FAILED; BOARD !. (GS2DERR 5215)
CAUSE
ACTION

5216 FASTCAT SIMULTANEOUS SPU/GPU TEST OF DRAM FAILED; BOARD !. (GS2DERR 5216)
CAUSE
ACTION

5217 FASTCAT SIMULTANEOUS SPU/GPU TEST OF DRAM AND SRAM FAILED; BOARD !. (GS2DERR 5217)
CAUSE
ACTION

5218 FASTCAT COMBINED SPU/GPU TEST OF SRAM FAILED; BOARD !. (GS2DERR 5218)
CAUSE
ACTION

5219 FASTCAT SIMULTANEOUS SPU/GPU TEST OF SRAM FAILED; BOARD !. (GS2DERR 5219)
CAUSE
ACTION

5220 FASTCAT Transform Engine TEST FAILED; BOARD !. (GS2DERR 5220)
CAUSE
ACTION

5221 FASTCAT GPU TEST FAILED; BOARD !. (GS2DERR 5221)
CAUSE
ACTION

5222 TRANSFORM TEST ERROR: RECTANGLE POINTS NOT INPUT PROPERLY. (GS2DERR 5222)
CAUSE
ACTION

5223 TRANSFORM TEST ERROR: INCORRECT NUMBER OF OUTPUT POINTS. (GS2DERR 5223)
CAUSE
ACTION

5224 TRANSFORM TEST ERROR: POLYGON DUMMY READ = ! SRC(!,!), DST(!,!). (GS2DERR 5224)
CAUSE
ACTION

5225 TRANSFORM TEST ERROR: MICROCODE REVISION READ !. (GS2DERR 5225)
CAUSE
ACTION

5226 Transform Engine FAILED: DEVICE COORDINATE MOVE. (GS2DERR 5226)
CAUSE
ACTION

5227 Transform Engine FAILED: DEVICE COORDINATE POLYLINE. (GS2DERR 5227)
CAUSE
ACTION

5228 Transform Engine FAILED: DEVICE COORDINATE POLYGON. (GS2DERR 5228)
CAUSE
ACTION

5229 Transform Engine FAILED: DEVICE COORDINATE POLYGON CLOSE. (GS2DERR 5229)
CAUSE
ACTION

5230 Transform Engine FAILED: DEVICE COORDINATE RECTANGLE. (GS2DERR 5230)
CAUSE
ACTION

5231 Transform Engine FAILED: HOLLOW DEVICE COORDINATE RECTANGLE. (GS2DERR 5231)

CAUSE

ACTION

5232 Transform Engine FAILED: DIRECT MOVE. (GS2DERR 5232)

CAUSE

ACTION

5233 Transform Engine FAILED: DIRECT DRAW. (GS2DERR 5233)

CAUSE

ACTION

5234 Transform Engine FAILED: CIRCLE. (GS2DERR 5234)

CAUSE

ACTION

5235 Transform Engine FAILED: WORLD COORDINATE MOVE. (GS2DERR 5235)

CAUSE

ACTION

5236 Transform Engine FAILED: WORLD COORDINATE POLYLINE. (GS2DERR 5236)

CAUSE

ACTION

5237 Transform Engine FAILED: WORLD COORDINATE POLYGON. (GS2DERR 5237)

CAUSE

ACTION

5238 Transform Engine FAILED: WORLD COORDINATE POLYGON CLOSE. (GS2DERR 5238)

CAUSE

ACTION

5239 Transform Engine FAILED: WORLD COORDINATE RECTANGLE. (GS2DERR 5239)
CAUSE
ACTION

5240 Transform Engine FAILED: HOLLOW WORLD COORDINATE RECTANGLE. (GS2DERR 5240)
CAUSE
ACTION

5241 Transform Engine FAILED: INVALID MICROCODE REVISION. (GS2DERR 5241)
CAUSE
ACTION

5242 Transform Engine FAILED: REGISTER WRITE. (GS2DERR 5242)
CAUSE
ACTION

5243 Transform Engine FAILED: REGISTER READ. (GS2DERR 5243)
CAUSE
ACTION

5244 Transform Engine FAILED: INCORRECT CURRENT POSITION. (GS2DERR 5244)
CAUSE
ACTION

5245 Transform Engine FAILED: TRANSFORM POINT. (GS2DERR 5245)
CAUSE
ACTION

5246 ERROR IN XSRC: SRC(!,!), DST(!,!); ACTUAL: !. (GS2DERR 5246)
CAUSE
ACTION

5247 ERROR IN YSRC: SRC(!,!), DST(!,!); ACTUAL: !. (GS2DERR 5247)
CAUSE
ACTION

5248 ERROR IN XDST: SRC(!,!), DST(!,!); ACTUAL: !. (GS2DERR 5248)
CAUSE
ACTION

5249 ERROR IN YDST: SRC(!,!), DST(!,!); ACTUAL: !. (GS2DERR 5249)
CAUSE
ACTION

5250 REJECTION ERROR: SRC(!,!), DST(!,!). (GS2DERR 5250)
CAUSE
ACTION

5251 GPU TEST NO. ! FAILED. (GS2DERR 5251) CRC SHOULD BE !; WAS !.
CHECKSUM SHOULD BE !; WAS !.
CAUSE
ACTION

5252 GPU ERROR; LEVEL = !; ERROR NO. = !. (GS2DERR 5252)
CAUSE
ACTION

5253 SPU MEMORY FAILURE: TEST #! ADDRESS 0x! EXPECTED 0x! ACTUAL 0x!.
(GS2DERR 5253)
CAUSE
ACTION

5254 GPU MEMORY FAILURE: TEST #! ADDRESS 0x! EXPECTED 0x! ACTUAL 0x!.
(GS2DERR 5254)
CAUSE
ACTION

6

98730A 3D Graphics Subsystem Diagnostic

Introduction

The 98730A Graphics Subsystem Diagnostic (GS3DDIAG) provides an on-line test of 98730A Graphics Subsystems on any Spectrum processor which supports the online diagnostics subsystem. The Graphics Subsystem can be configured as the system console or as a graphics peripheral. The type of tests that are provided can test all functions of the 98730A Graphics Subsystem.

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10018.

Minimum Configuration

GS3DDIAG is designed to test and verify 98730A Graphics Subsystems configured as system consoles or peripherals for any PA-RISC machine. The system must be up and running. and have 1 of 3 possible graphics interfaces correctly configured and installed.

The following three assemblies are always required, although they are not used in all sections (all tests that produce visual output will require a functional Color Map):

- Frame Buffer Controller (part no.98730-66571)
- Color Map (part no. 98730-66575)
- Frame Buffer 1 (part no. 98720-66572)

The following optional assemblies are required for specific test sections only, as indicated in the discussion of those test sections which appears in "Section Functional Descriptions." below.

- Frame Buffer 2 and 3 (part no. 98720-66572)
- Fast Z Buffer (part no. 98730-66573)
- Master (needs Fast Z) (part no. 98730-66577)
- Slave (needs Master) (part no. 98730-66579)

The first graphics interface is the A1017A mid_bus to LGB interface card which may reside in mid_bus slot 2, 3, 4, or 5 . One to four A1017A's can be connected to the mid_bus in these slots, so that up to four 98730A's may be tested simultaneously.

The diagnostic will also work with an A1047A graphics interface. The A1047A consists of a full length mid_bus/DMA card (98730-66583) and a half length CIO to LGB card (98730-66582) connected by a ribbon cable (the cards must be in adjacent slots). An additional 98730-66582 (A1048A) may be installed adjacent to the first 98730-66582, connected to the same ribbon cable. One 98730A may be connected to each 98730-66582, so that up to two 98730A's may be configured per system. Two 98730A's cannot be tested simultaneously though, and a system may not mix A1017A's a A1047A's. Please see the *Service Familiarization Guide* for further configuration details.

Operating Instructions

The 98730A Graphics Subsystem Diagnostic can be accessed by the user via the Diagnostic User Interface.

Default Tests

If the user does not specify which section to run, the default is to run all of the test sections. except Section 40.

RUN Command

To run this diagnostic, enter:

```
DUI >run gs3ddiag pdev=8.0 /dev/crtxx <RUN Command Options>
                        |         |
                        |  none required for
                        |  default test suite
                        |
      The device= parameter must be used and
      it must point to one of 4 device files:
               crt0  crt1  crt2  crt3
```

GS3DDIAG may be run from the 98730A being tested or from a remote terminal. In either case, no keyboard input should be entered on the 98730A, since this can corrupt the current section being run. Also. the mouse should be disconnected to prevent input from it. It is suggested that the diagnostic be run from a remote terminal so that messages will appear on the remote terminal and test patterns will appear on the 98730A. If the diagnostic is run from the 98730A being tested. the section will finish. refresh the text screen, pause, and then run the next section.

As usual. a software termination signal (usually invoked with a CNTL-C) will terminate GS3DDIAG imediately in almost all situations. The only exception is when certain pictures are being downloaded. There will be a short delay before CNTL-C will work.

The **device=** parameter must be used and it must point to one of 4 device files: **crt0**. **crt1**. **crt2**. or **crt3**. These correspond to the **lu** of the graphics interface from the gen file. With an A1017A. any of the four might be used. With an A1047A. "crt0" would be used; while with an A1048A, either "crt0" or "crt1" would be used. Please refer to the *HP9000/825/835 System Administrator Guide* for more details on device files.

Note if the **erronly** parameter is set "on". only error messages will be output by the diagnostic. Error messages can be distinguished from other messages by three "*"'s preceding the text of the message (i.e., '*** MESSAGE' is an error message whereas 'Message' is not). Also. note that error messages are in capital letters and other messages use some lower case characters. As noted in the previous section. all tests run in destructive mode.

Test Execution

When GS3DDIAG is run, a header and welcome message will be displayed.

At this point the diagnostic prompts the user with the following questions:

```
Are you testing the 98730A from a different terminal? (y/n) [n]

Do you want this program to determine the unit configuration?  (y/n) [y]
```

With the default (YES) the program will attempt to identify the hardware configuration of the 98730A. If this procedure fails, it will return to the question "**Do you want this program to determine the hardware configuration?**"

If you answer no to the above question, the following questions will be asked:

```
Enter the number of 98720-66572 Frame Buffer boards: (a=1/b=2/c=3) [c]

Is 98730-66573 Fast Z Buffer board present? (y/n) [y]

Is 98730-66577 Master board present? (y/n) [y]

Is 98730-66579 Slave board present? (y/n) [y]
```

The program will then return a combination of messages stating the IODC revisions and which boards are present or absent.

```
Do you wish to continue with this unit configuration?  (y/n) [y]
```

Note

If the user runs GS3DDIAG from the graphics subsystem as the system console, CTRL -C should be pressed to interrupt the diagnostic; the CTRL SHIFT RESET keys should also be pressed to reset the graphics subsystem, and to allow GS3DDIAG to return to the DUI> prompt.

After answering the questions, the sections specified by the user will be executed and the results output. The default is to run all sections. There are no steps within any sections. If at any time, the number of errors generated reaches the limit specified by the user (10 for instance) in the ERRCOUNT parameter of the run command the following message will be output:

```
Error count exceeded. Count = 10
```

The diagnostic will then terminate execution. If the TRACE parameter is used, and when equated to PROGRAMS, it will generate informational messages. If the diagnostic is run on a 98730A being used as system console, then the TRACE info should be sent to a file to avoid corrupting visual output.

If the ERRPAUSE parameter of the run command was assigned to a value of "on", then this diagnostic will stop after each error is generated and ask the user if the test should continue:

```
Error pause -- do you wish to continue ([y]/n)?
```

If the response is "Y", then the test will be resumed (if possible), and if the response is "N", then this diagnostic will terminate. The "[Y]" indicates that "Y" will be the default response if the user simply hits <CR> in response to the prompt. If the sections specified by the user were executed the number of times specified in the LOOP parameter of the run command without the number of errors exceeding the ERRNUM value, the diagnostic will terminate normally. Upon termination of this diagnostic, control will return to the diagnostic system.

Detailed Test Descriptions

GS3DDIAG has the following available test sections:

Section 10:	Installation
Section 20:	Series 800 Interfaces
Section 21:	Frame Buffer Controller Logic
Section 22:	Strip 7 Buffer Logic
Section 23:	Fast 7 Buffer Logic
Section 24:	Frame Buffer Board 0 RAM via 1 GB
Section 25:	Frame Buffer Board 1 RAM via 1 GB
Section 26:	Frame Buffer Board 2 RAM via 1 GB
Section 27:	Fast 7 Buffer RAM
Section 28:	Overlay Buffer RAM
Section 29:	Color Map Tests
Section 30:	Master Board Logic
Section 31:	Scan Conversion
Section 32:	Transform Engine(s) Tests
Section 40:	Visual Tests (This is not a default section)

Section Functional Descriptions

The remainder of this document is devoted to describing each section of GS3DDIAG. Please note that in regard to the error messages, all possible error messages that may be generated are not listed. Times are listed in parenthesis which represent an estimate of the time it takes to run the section successfully on an unloaded 825 with an A1047A interface.

Section 10: Installation

This section tests some portions of all assemblies present. It is designed as a quick verification of the graphics subsystem functionality at installation time.

Additional required hardware:

Master Board - (part number 98730-66577)

Output:

Starbase Demo Torus	A rotating white torus.
Starbase Demo VW	A rotating white Volkswagen body.
Starbase Demo Spring	A white spring
Starbase Demo Glass1	A rotating white wine glass.

Section 20: Series 800 Interfaces

This section will test the DMA transfer capabilities on the A1047A interface when available.

Additional required hardware:

Nimbus Interface - (part numbers 98730-66583, 98730-66582)

Output:

There is no output for the A1047A Self Test, A1048A Self Test, A1047A DMA FB Addr, A1047A DMA Width.

ERIC DMA Synth	A white screen.
A1047A DMA Synth	A wavey star mosaic pattern which flows from upper left to lower right.

Section 21: Frame Buffer Controller Logic

This section tests logic circuitry residing on the Frame Buffer Controller, Frame Buffer(s), Fast Z Assembly, and Color Map.

Additional required hardware:

Fast Z - (part number 98730-66573)

Output:

There is no output for the ID/Font ROM, Frame Buffer Folded/Full. Frame Buffer Write Enable, or Frame Buffer Control Shadow RAM tests.

Fast Window Move	Output consists of a double black cross surrounded by a white border constructed of 128 bit wide lines. With one frame buffer the image will appear blue and white, and with two it will appear magenta.
Slow Window Move	Output consists of two rows of 32 grey shaded polygons at the upper half of the screen. In the second pass they appear to be filled with a random sequence of dots. The pattern will resemble a Moire pattern. Then a random mish-mash of white lines and polygons are displayed.

Section 22: Strip Z Buffer Logic

This section tests Z buffer logic on the Frame Buffer Controller, and Frame Buffer(s). It also utilizes some circuitry on the Master Board.

Additional required hardware:

Master Board - (part number 98730-66577)

Output:

Z Mapping Using Strip Z	A row of intersecting pairs of shaded quadrilaterals at top of screen and one pair of overlapping (non-intersecting) shaded quadrilaterals in second row. This is followed by white background with yellow rectangular on-screen Z buffer data for off-screen quadrilaterals. The color will vary with the number of framebuffers.
ZKEEP Using Strip Z	One pair of intersecting shaded quadrilaterals and one pair of overlapping shaded quadrilaterals.
ZEQ Using Strip Z	Two pairs of intersecting shaded quadrilaterals (intersection areas will appear different).
ZPASS Using Strip Z	A very fast flash.

Section 23: Fast Z Buffer Logic

This section tests the Fast Z Buffer logic on the Frame Buffer Controller and the Fast Z assemblies. It also utilizes some of the circuitry on the Frame Buffer(s) and Master assemblies.

Additional required hardware:

- Fast Z - (part number 98730-66573)
- Master Board - (part number 98730-66577)

Output:

Z Mapping Using Fast Z	A row of intersecting pairs of shaded quadrilaterals at top of screen and one pair of overlapping (non-intersecting) shaded quadrilaterals in second row.
ZKEEP Using Fast Z	One pair of intersecting shaded quadrilaterals and one pair of overlapping shaded quadrilaterals.
ZEQ Using Fast Z	Two pairs of intersecting shaded quadrilaterals (intersection areas will appear different).
ZPASS Using Strip Z	A very fast flash.

Section 24: Frame Buffer Board 0 RAM via LGB

This section tests the Frame Buffer Board 0 RAM via the LGB Bus.

Output:

Output consists of a blue raster drawn from top to bottom, then erased from top to bottom, then drawn from bottom to top, then erased from bottom to top and finally a fast draw from top to bottom.

Section 25: Frame Buffer Board 1 RAM via LGB

This section tests the Frame Buffer Board 1 RAM via the LGB Bus.

Additional required hardware:

Frame Buffer 1 - (part number 98720-66572)

Output:

Output consists of a green raster drawn from top to bottom, then erased from top to bottom, then drawn from bottom to top, then erased from bottom to top and finally a fast draw from top to bottom.

Section 26: Frame Buffer Board 2 RAM via LGB

This section tests the Frame Buffer Board 2 RAM via the LGB Bus.

Additional required hardware:

Frame Buffer 1 and 2 - (part number 98720-66572)

Output:

Output consists of a red raster drawn from top to bottom, then erased from top to bottom, then drawn from bottom to top, then erased from bottom to top and finally a fast draw from top to bottom.

Section 27: Fast Z Buffer RAM

This section tests the RAM on the Fast Z Board.

Additional required hardware:

Fast Z - (part number 98730-66573)

Output:

There is no output for the Frame Buf FZU Ram Via LGB, and Frame Buf FZL Ram Via LGB tests.

Section 28: Overlay Buffer RAM

This section tests the Overlay RAM on the Color Map Assembly.

Output:

There is no output for the Overlay RAM via LGB test.

Color Map CRC registers	Output consists of a herringbone pattern across the entire screen.

Section 29: Color Map Tests

This section tests the Color Map assembly.

Output:

There is no output for the CMap ALU. CMap Shadow RAM, CMap IRIS. and CMap IRIS Registers tests.

Color Map Crosshair CRC	A spiral moving red crosshair.
Color Map Rband Line CRC	A red sweeping line like a clock second hand.
Color Map Rband Rect CRC	A moving red rectangle.
Color Map Sprite CRC	A moving yellow happy face.
Cmap Power Up	A white flash.

Section 30: Master Board Logic

This section tests the logic circuitry on the Master Board.

Additional required hardware:

Master Board - (part number 98730-66577)

Output:

There is no output for the Xform ID. LGBDAT, LGBREG. LGBADD. DC RAM over LGB. Ace Registers. DC RAM via ACE, Xform Semaphore Register. DCRAM_x via master treis. DCRAM_y via master treis, and Pace Registers test.

LGB Grant	See section 21 Slow Window Move test.

Section 31: Scan Conversion

This section tests the Scan Conversion circuitry (ACE & PACE) on the Master Board. It also uses some circuitry on the Frame Buffer Controller and the Frame Buffer(s).

Additional required hardware:

Master Board - (part number 98730-66577)

Output:

There is no output for the Pace Bounding test.

Ace test	Output consists of a shaded polygon covering the full screen with green at the lower left, blue lower right, red upper right and white upper left. Then three small multi-colored polygons below what looks like the outline of mountain peaks on a mosaic plane with a line of grey bars below it. Finally a red, green, and blue triangle appears in the upper left.
Pace Clipping	Output consists of a blue rectangle at about the center of the screen, then a red rectangle partially behind the blue, then a green partially behind both, then a magenta behind all three and finally a white rectangle behind all of them.
Pace Dither	Output consists of four polygons on the screen. The lower right and upper left are shaded with blue lower left, white lower right, green upper right and red upper left. The lower left and upper right polygons are grey shaded with white at upper left and lower right and black at lower left and upper right.
Pace Gamma	Output consists of two polygons at the upper left and lower right. Both have blue at lower left, white at lower right , green upper right and red upper left. Color intensity of the lower right polygon will appear brighter than the upper left.
Pace Transparency	Output consists of two shaded polygons at the upper left and lower right. Both polygons have blue lower left, white lower right, green upper right and red upper left. Both polygons will appear slightly dim.
Pace Valve	Output consists of a shaded rectangle covering the entire screen with green lower left, blue lower right, red upper right and white upper left.

Section 32: Transform Engine(s) Tests

This section tests the Transform Engines on the Master and Slave Boards.

Additional required hardware:

Master Board - (part number 98730-66577) Slave Board (optional) - (part number 98730-66579)

Output:

There is no output for the following tests:

- Xform Random Control Store
- Xform IEEE Control Store
- Cntrl Store Walking Bit
- Xform Floating Point
- PRAM
- Dcinf Master/Slave
- Engine 1 CDRAM Bank X
- Engine 1 CDRAM Bank Y
- Engine 2 CDRAM Bank X
- Engine 2 CDRAM Bank Y
- Engine 3 CDRAM Bank X
- Engine 3 CDRAM Bank Y
- DCRAM_X via Master Treis
- DCRAM_Y via Master Treis
- Mstr/Slv DATARAM via Treis
- Mstr/Slv CDRAM_X via Treis
- Mstr/Slv CDRAM_Y via Treis

- Treis Sequencer Registers
- Treis Return Stack
- Treis Addr Gen & ALU Reg
- Treis ALU Register File 1
- Treis ALU Register File 2
- Treis Download/Upld Ucode
- Treis Unique Reg Address
- Treis Interrupt Instr
- Treis Stack Overflow
- Treis Stack Underflow
- Treis Loop Instructions
- Treis Jump Instructions
- Treis Jmp Sub & Rtn Instr
- Treis IDB Source and Dest
- Treis ALU Source and Dest
- Treis ALU Function & Stat
- Treis ALU A/B Addr Decode
- Treis Pointer RAM
- Treis PRAM Selftest
- Treis Address Generator
- Treis Ready Condition Code
- Treis Q Register Rotate
- Treis FP Funct/Stat CCodes

- Treis F/S: Add Scalar Mode
- Treis F/S: Add Tpipe1 Mode
- Treis F/S: Add Tpipe2 Mode
- Treis F/S: Add Tpipe3 Mode
- Treis F/S: Add Tpipe4 Mode
- Treis F/S: Add Tpipe5 Mode
- Treis F/S: Mul Scalar Mode
- Treis F/S: Mul Tpipe1 Mode
- Treis F/S: Mul Tpipe2 Mode
- Treis F/S: Mul Tpipe3 Mode
- Treis F/S: Mul Tpipe4 Mode
- Treis F/S: Mul Tpipe5 Mode
- Treis F/S: Div Scalar Mode
- Treis F/S: Div Tpipe1 Mode
- Treis F/S: Div Tpipe2 Mode
- Treis F/S: Div Tpipe3 Mode
- Treis F/S: Div Tpipe4 Mode
- Treis F/S: Div Tpipe5 Mode
- Treis MW Addressing PRAM

Section 40: Visual Tests

This section invokes a set of visual tests that generate images that reveal problems with the color map circuitry. The color map hardware converts frame buffer image data into high speed analog signals capable of driving a color monitor. It is very difficult to test this hardware directly so these images must be interpreted by an operator to determine if a problem exists. The intent of the majority of the displays is obvious and should require no explanation.

Additional required hardware:

None.

Output:

The following images will appear. The colors will vary depending on the number of frame buffers present.

Cmap Zoom	A set of sixteen white squares are turned red, one at a time.
Cmap Pan	Inside a red border, a blue band moves horizontally across screen, then vertically up screen.
Cmap Blink	A blinking red square inside a non-blinking green square inside a non-blinking blue square.
Cmap Blink Overlays	One non-blinking red square and three blinking red squares in a black, white, green, and light blue colored column.
Cmap Vdrive Cad	Blue square and single blue rectangle. Next, a green square and two green rectangles. Next, a red square and three red rectangles. Next, a white square and one blue, two green and three red rectangles.
Cmap Vdrive Imaging	Three Overlapping red opaque squares.
Cmap Match	Three red squares placed to make a red column. Next, three green squares placed to make a green column. Next, three blue squares placed to make a blue column. Then the columns are dimmed.
Cmap Refresh Stuck Low	Four red-green-blue dashed vertical lines.

Cmap Refresh Stuck High	Four red-green-blue dashed vertical lines.
Cmap Crosshair	Red crosshair traveling along a white spiral turning the spiral green.
Cmap Rband Line	A white circle turned blue by a red sweep line. Looks like a clocks second hand.
Cmap Rband Rect	A shrunken red square traveling along a white line pattern, turning the line blue.
Cmap Sprite	A Yellow smiles face traveling along a white line pattern, turning the white line pattern purple. At the end of the pattern, the smiles face turns to a devil face with blinking white eyes and a black mouth.
Cmap 1-k Mod	A red square, blue square, red square, and the red square is changed to a blue square from right to left.
Visual Fast Window Move	See section 21 Fast Window Move output
Visual Slow Window Move	See section 21 Slow Window Move output.
Visual Pace Clipping	See section 31 Pace Clipping.
Visual Pace Dither	See section 31 Pace Dither.
Visual Pace Transparency	See section 31 Pace Transparency.
Visual Pace Gamma	See section 31 Pace Gamma.
Visual Pace Valve	See section 31 Pace Valve.

Information Messages

```
INVALID REPLY
```

```
Error count exceeded.  Count = !
```

```
Error pause -- do you wish to continue? (y/n) [y]
```

```
Are you testing the 98730A from a different terminal? (y/n) [n]
```

```
Do you want this program to determine the unit configuration?  (y/n) [y]
```

```
Enter the number of 98720-66572 Frame Buffer boards: (a=1/b=2/c=3) [c]
```

```
Is 98730-66573 Fast Z Buffer board present? (y/n) [y]
```

```
Is 98730-66577 Master board present? (y/n) [y]
```

```
Is 98730-66579 Slave board present? (y/n) [y]
```

```
IODC HVERSION: MODEL !  Rev !
```

```
IODC SVERSION: MODEL !  Rev !  Opt !
```

```
IODC      Rev: !
```

```
98730-66573 Absent   (Fast Z Buffer)
```

```
98730-66573 Present  (Fast Z Buffer)
```

```
98720-66572 Absent   (Frame Buffer !)
```

```
98720-66572 Present  (Frame Buffer !)
```

```
98730-66577 Absent   (Master)
```

```
98730-66577 Present  (Master)
```

```
98730-66579 Absent   (Slave)
```

```
98730-66579 Present  (Slave)
```

```
98730-66575 Present  (High-Performance Color Map)
```

```
Do you wish to continue with this unit configuration?  (y/n) y
```

```
Loop count = !
```

```
Begin Section !  - !
```

```
End of Section !  - !
```

FRU Table

The following table summarizes results of the diagnostic sections. Each of the entries constitutes a separate message, which may be printed. Here is the key to table entries:

[fff] Testing this FRU was a major goal (section FAILED)

[ppp] Testing this FRU was a major goal (section PASSED)

[f] Exercising a portion of this FRU was necessary to achieve the section goal (section FAILED)

[p] Exercising a portion of this FRU was necessary to achieve the section goal (section PASSED)

[-] This FRU was not exercised

FRU Tested:	**IF**	**FBC**	**FB0**	**FB1**	**FB2**	**FZ**	**MST**	**SLV**	**CM**
Result:									
Section ! (no data)									
Section 10 Passed	ppp	ppp	ppp	ppp	ppp	ppp	ppp	ppp	-
Section 10 Failed	fff	fff	fff	fff	fff	fff	fff	fff	-
Section 20 Passed	ppp	-	-	-	-	-	-	-	-
Section 20 Failed	fff	-	-	-	-	-	-	-	-
Section 21 Passed	p	ppp	ppp	ppp	ppp	ppp	-	-	ppp
Section 21 Failed	f	fff	fff	fff	fff	fff	-	-	fff
Section 22 Passed	p	ppp	ppp	ppp	ppp	p	p	-	p
Section 22 Failed	f	fff	fff	fff	fff	f	f	-	f
Section 23 Passed	p	ppp	ppp	ppp	ppp	ppp	p	-	p
Section 23 Failed	f	fff	fff	fff	fff	fff	f	-	f
Section 24 Passed	p	ppp	ppp	p	p	p	-	-	p
Section 24 Failed	f	fff	fff	f	f	f	-	-	f
Section 25 Passed	p	ppp	p	ppp	p	p	-	-	p
Section 25 Failed	f	fff	f	fff	f	f	-	-	f
Section 26 Passed	p	ppp	p	p	ppp	p	-	-	p
Section 26 Failed	f	fff	f	f	fff	f	-	-	f
Section 27 Passed	p	ppp	p	p	p	ppp	-	-	p
Section 27 Failed	f	fff	f	f	f	fff	-	-	f
Section 28 Passed	p	ppp	p	p	p	p	-	-	ppp
Section 28 Failed	f	fff	f	f	f	f	-	-	fff
Section 29 Passed	p	-	-	-	-	-	-	-	ppp
Section 29 Failed	f	-	-	-	-	-	-	-	fff
Section 30 Passed	p	-	-	-	-	-	ppp	-	-
Section 30 Failed	f	-	-	-	-	-	fff	-	-
Section 31 Passed	p	p	p	p	p	p	ppp	-	p
Section 31 Failed	f	f	f	f	f	f	fff	-	f
Section 32 Passed	p	-	-	-	-	-	ppp	ppp	-
Section 32 Failed	f	-	-	-	-	-	fff	fff	-
Section 40 Passed	p	p	p	p	p	p	p	p	ppp
Section 40 Failed	f	f	f	f	f	f	f	f	f

Error Messages

The following error messages are arranged in numerical order. CAUSE messages indicate which sections have failed. EFFECT messages should be used, in conjunction with the FRU Table, to identify the most likely FRU.

CAUSE: These messages indicate which sections have failed.
EFFECT: Use this information along with the FRU Table to identify the most likely FRU

89	*** ERROR OCCURRED DURING INSTALLATION TEST (GS3DERR 89)
90	*** ERROR OCCURRED DURING S800 INTERFACE TESTING (GS3DERR 90)
91	*** ERROR OCCURRED DURING FRAME BUFFER CONTROLLER LOGIC TESTING (GS3DERR 91)
92	*** ERROR OCCURRED DURING STRIP Z BUFFER LOGIC TESTING (GS3DERR 92)
93	*** ERROR OCCURRED DURING FAST Z BUFFER LOGIC TESTING (GS3DERR 93)
94	*** ERROR OCCURRED DURING FRAME BUFFER BOARD 0 RAM TESTING (GS3DERR 94)
95	*** ERROR OCCURRED DURING FRAME BUFFER BOARD 1 RAM TESTING (GS3DERR 95)
96	*** ERROR OCCURRED DURING FRAME BUFFER BOARD 2 RAM TESTING (GS3DERR 96)
97	*** ERROR OCCURRED DURING FAST Z BUFFER RAM TESTING (GS3DERR 97)
98	*** ERROR OCCURRED DURING OVERLAY BUFFER RAM TESTING (GS3DERR 98)
99	*** ERROR OCCURRED DURING COLOR MAP LOGIC TESTING (GS3DERR 99)
100	*** ERROR OCCURRED DURING MASTER BOARD LOGIC TESTING (GS3DERR 100)
101	*** ERROR OCCURRED DURING SCAN CONVERSION TESTING (GS3DERR 101)
102	*** ERROR OCCURRED DURING TRANSFORM ENGINE TESTING (GS3DERR 102)
103	*** ERROR OCCURRED DURING VISUAL (INTERACTIVE) TESTING (GS3DERR 103)

CAUSE: These messages indicate that communication with the Device Under test cannot be achieved.

EFFECT: This can occur under the following circumstances:

- Device is not turned on.
- Device was not powered up and connected to the SPU when the SPU powered up.
- The wrong device file is being used.
- Another test process already owns the device.
- The device files have been installed incorrectly or have become corrupted.
- Graph2 has not been installed in your kernel. See uxgen.

---

104 `*** CANNOT OPEN DIAG DEV FILE.  ! <== OPEN('!').   errno = ! (GS3DERR 104)`

---

105 `*** CANNOT GCDESCRIBE DIAG DEV FILE '!'. (GS3DERR 105)`

---

106 `*** CANNOT GCMAP DIAG DEV FILE '!'. (GS3DERR 106)`

---

107 `*** CANNOT MAP UNIT. (GS3DERR 107)`

---

108 `*** CHECK OF DEVICE FAILED. (GS3DERR 108)`

---

109 `*** MAP OF DEVICE FAILED. (GS3DERR 109)`

CAUSE: These messages indicate that the user has given the diagnostic invalid information pertaining to the hardware or diagnostic configuration or the hardware has been organized in an unsupported configuration.

EFFECT: Check the documentation to see that all information and hardware installation is valid and supported.

---

110 `*** UNSUPPORTED HARDWARE CONFIGURATION. (GS3DERR 110)`

CAUSE: Miscellaneous self explanatory messages.

EFFECT: NONE.

---

111 `*** DAVINCI TEST PROGRAM ABORTED (GS3DERR 111)`

CAUSE: Detailed trace messages. These should only be examined if the FRU Selection table provided unsatisfactory results.

EFFECT: These messages are availible when the `DUI` "trace=prog" option is selected. These messages require a thorough understanding of the hardware and test internals to properly interpret.

---

112 `*** CAUGHT BUS ERROR. (GS3DERR 112)`

---

113 `*** CAUGHT TERMINATION SIGNAL. (GS3DERR 113)`

CAUSE: These error messages usually indicate that the diagnostic has been improperly installed onto the HP-UX system or that certain files have been deleted or corrupted.

EFFECT: Re-install online diagnostics. If the problem persists then call the factory.

---

114 `*** ERROR. CALL_SB(): CANNOT OPEN EXTERNAL PROGRAM. MUST OPEN IN ORDER TO LOCK. (GS3DERR 114)`

---

115 `*** (PARENT PROCESS) 'THE_DEMO' UN-EXECUTABLE.`
`CHECK KERNEL PROCESS LIMIT. (GS3DERR 115)`

---

116 `*** ERROR OPENING TEXT FILE. (GS3DERR 116)`

---

117 `*** CAN'T OPEN CRC REFERENCE FILE '!' (GS3DERR 117)`

---

118 `*** CRC NUMBER !. BOARD CRC ON BOARD !. (GS3DERR 118)`

---

119 `*** ERRNO = ! (GS3DERR 119)`

---

120 `*** ERROR READING FILE HEADER. ! <== READ() (GS3DERR 120)`

---

121 `*** INVALID MAGIC FIELD. PROBABLY WRONG FILE TYPE. (GS3DERR 121)`

---

122 `*** ERROR READING SEGMENT HEADER. (GS3DERR 122)`

---

123 `*** UNSUPPORTED SEGMENT TYPE: "!" (GS3DERR 123)`

---

124 `*** ERROR READING UCODE DATA. (GS3DERR 124)`

125	`*** CAN'T CLOSE ! (GS3DERR 125)`
126	`*** CANNOT LOCK FILE '!'. THIS FILE MUST BE LOCKED TO INSURE THAT ONLY ONE TEST PROCESS IS EXECUTING THIS TEST AT ONE TIME. THIS IS REQUIRED BECAUSE THIS TEST CONSUMES ALL AVAILIBLE SYSTEM MEMORY. (GS3DERR 126)`
127	`*** CONTROL SPACE DEVICE FILE CANNOT BE LOCKED. ERRNO = ! (GS3DERR 127)`

128	`*** ERROR: RESOURCE CANNOT BE UNLOCKED.`
	`FILE: "!"`
	`ERRNO: ! (GS3DERR 128)`

129	`*** ERROR: RESOURCE CANNOT BE LOCKED. LOCK_FILE()`
	`FILE: "!"`
	`ERRNO: !`
	
	`POSSIBLE CAUSES:`
	`- FILE DESCRIPTOR WAS NEVER OPENED.`
	`- FILE WAS NOT OPENED FOR O_RDWR OR O_WRONLY. (GS3DERR 129)`

130	`*** CAN'T OPEN "!". ! <== OPEN() (GS3DERR 130)`
131	`*** CAN'T FIND OR ACCESS 'ETC' DIRECTORY. (GS3DERR 131)`

CAUSE: These error messages occur when the hardware defect prevents the requested operation from continuing.

EFFECT: Change you strategy to one of the following: - Using an explicit configuration rather than relying on the diagnostic auto-configuring.

- Network in and test the device remotely rather than testing a device who is simultaneously operating as an ITE.

- Diagnose the problem "Off Line" with ISL or Self Test (if available).

132	`*** ITE STATE CANNOT BE READ. BAD FBC OR COLOR MAP. (GS3DERR 132)`
133	`*** REPORT_INTERFACE_CONFIG(): CANNOT READ IO_STATUS() REG. BAD INTERFACE.` `(GS3DERR 133)`
134	`*** CAN NOT READ 98730A TERMINAL STATE DUE TO HARDWARE FAILURE. (GS3DERR 134)`

135	*** ERROR TRYING TO DETERMINE UNIT CONFIGURATION. (GS3DERR 135)
136	*** BUS ERROR WHILE EXTRACTING COLOR MAP STATE. (GS3DERR 136)
137	*** BUS ERROR WHILE EXTRACTING FRAME BUFFER CONTROLLER STATE. (GS3DERR 137)
138	*** ERROR TRYING TO DETERMINE DAVINCI CONFIGURATION. (GS3DERR 138)
139	*** USE EXPLICIT CONFIGURATION ARGUMENT. (GS3DERR 139)
140	*** CHECK TO SEE IF UNIT POWER IS ON. THEN, CONTINUE BY SPECIFYING THE UNIT CONFIGURATION MANUALLY. (GS3DERR 140)

CAUSE: These error messages usually indicate that a system resource has been exhausted. This can occur when the system is heavily loaded or when ram, swap space or disk space is limited.

EFFECT: Investigate the state of the system and correct the resouce limitation. Some tools and areas to consider are:

- Use the utility "monitor", "df" and "du" to determine the file system and swap space utilization.

- Check and see if the sections that report the message require root capability.

- Reduce the number of processes currently running on the system. See the "ps" command.

- Add more ram memory. When interfaces that support DMA are installed, substantial amounts of lockable memory may be required for the diagnostic and the user application.

141	*** ERROR <== FDOPEN() IN DUMP_PIPE_NLSL(). ERRNO = ! POSSIBLE CAUSES: - FILE SYSTEM FULL (GS3DERR 141)
142	*** ERROR <== PIPE(). ERRNO = ! POSSIBLE CAUSES: - FILE SYSTEM FULL (GS3DERR 142)

143 *** FORK() FAILED. ERRNO = ! (GS3DERR 143)

---

144 *** CAN'T ALLOCATE DYNAMIC MEMORY FOR UCODE. (GS3DERR 144)

---

145 *** ERRNO = ENOMEM = ! (GS3DERR 145)

---

146 *** PROBABLY OUT OF SWAP SPACE. (GS3DERR 146)

---

147 *** OLD TOP OF HEAP = OX! <== SBRK(0). (GS3DERR 147)

---

148 *** NEW TOP OF HEAP = OX! <== SBRK(0) + UCODE REQUIREMENT. (GS3DERR 148)

---

149 *** CANNOT ALLOCATE SPU MEMORY FOR PICTURE USING MALLOC().

POSSIBLE CAUSES:
- NOT ENOUGH RAM OR SWAP SPACE.
- TOO MANY PROCESSES. (GS3DERR 149)

---

150 ***CANNOT DATLOCK DATA SEGMENT INTO MEMORY.

POSSIBLE CAUSES:
- NOT ENOUGH RAM OR SWAP SPACE.
- TOO MANY PROCESSES.
- NOT SUPER USER.
- A LOCK ALREADY EXISTS ON THE DATA SEGMENT. (GS3DERR 150)

CAUSE: These error messages should never occur. They are built in to the diagnostic to help the diagnostic developer catch obvious bugs. They indicate a defect in the diagnostic.

EFFECT: Gather as much information together about the conditions that caused this message to be reported and notify the factory.

---

151 *** ERROR. TEST_FP(): INVALID RETURN VALUE FROM OBJ_DLOAD(). (GS3DERR 151)

---

152 *** ERROR READING TEXT FILE. (GS3DERR 152)

153	*** WRITE_CMAP_VALUE(): UNSUPPORTED CMAP SELECTOR (GS3DERR 153)
154	*** DUMP_SYSTEM_CRC(): BD OUT OF RANGE. (GS3DERR 154)
155	*** CRC_GETS(): PREMATURE EOF WHILE READING CRC REFERENCE FILE. (GS3DERR 155)
156	*** ERROR ENCOUNTERED WHILE READING CRC REFERENCE FILE. (GS3DERR 156)
157	*** FILE: "!" (GS3DERR 157)
158	*** BAD TOKEN: "!" (GS3DERR 158)
159	*** CRC NUMBER !. RAM CRC ON BOARD !. PLANE !. RAM !. (GS3DERR 159)
160	*** READ_BDL_CRC_UCODE(): UNRECOGNIZED VALUE <== OBJ_DLOAD() (GS3DERR 160)
161	*** OBJ_DLOAD(): UNSUPPORTED ENGINE SELECTOR '!' (GS3DERR 161)
162	*** OBJ_DLOAD(): INVALID UCODE SELECTOR NUMBER. (GS3DERR 162)
163	*** ERROR. ALLOCATE_MEMORY(): NUM_COLUMNS PARAMETER OUT OF RANGE. (GS3DERR 163)
164	*** ERROR. ALLOCATE_MEMORY(): NUM_LINES PARAMETER OUT OF RANGE. (GS3DERR 164)
165	*** CANNOT UNLOCK DATA SEGMENT IN MEMORY. POSSIBLE CAUSES: - NO SEGMENT LOCK IS IN EFFECT. - NOT SUPER USER. (GS3DERR 165)
166	*** ERROR. INVALID WALKING BIT SEQUENCE DESIGNATOR = ! (GS3DERR 166)
167	*** RESET_HARDWARE(): UNKNOWN INTERFACE. (GS3DERR 167)
168	*** LOCK_FRAME(): FRAME BUFFER DEVICE FILE NEVER OPENED. (GS3DERR 168)

169	*** (UN)LOCK_FRAME(): UNSUPPORTED INTERFACE (GS3DERR 169)
170	*** ERROR. ARM_SIGNAL_TRAPPING(): SHERLOCK SIGNAL HANDLING IS NOT IMPLEMENTED HERE AND SHOULD BE INSTALLED IN GS3DDIAG.C. (GS3DERR 170)
171	*** WARNING: "!" SYSTEM CALL IN TEST INTERRUPTED. (GS3DERR 171)
172	*** WARNING: SIGNAL NUMBER ! TRAPPED. (!) (GS3DERR 172)
173	*** ENABLE_SIGNALS(): CALLED BEFORE DISABLE_SIGNALS(). (GS3DERR 173)
174	*** ERROR: ARM_SIGNAL_TRAPPING(): UNSUPPORTED COMMAND NAME. (GS3DERR 174)
175	*** ERROR: TEST_PAUSE(): ATTEMPTING TO ALARM() W SIGNALS DISABLED. UTIL_TESTPAUSE_DL TEST_PAUSE(): ATTEMPTING TO ALARM() W SIGNALS DISABLED. (GS3DERR 175)
176	*** ERROR: CTS(): INVALID NEW_FLAG = ! (GS3DERR 176)
177	*** ERROR: CTS(): INVALID DEST_FLAG = ! (GS3DERR 177)
178	*** INIT_WALK(): UNSUPPORTED SEQUENCE SELECTOR. (GS3DERR 178)
179	*** READ_ENGINE(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 179)
180	*** WRITE_ENGINE(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 180)
181	*** SAVE_TREIS_PDR(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 181)
182	*** SAVE_TREIS_PAR(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 182)
183	*** SAVE_TREIS_PAD(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 183)
184	*** RESTORE_TREIS_PDR(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 184)
185	*** RESTORE_TREIS_PAR(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 185)

186	*** RESTORE_TREIS_PAD(): INVALID ENGINE SELECT CHAR '!' (GS3DERR 186)
187	*** WAIT_HALT_SET(): TIME_LIMIT MUST BE GREATER THAN 1 SEC. (GS3DERR 187)
188	*** REPORT_INTERFACE_CONFIG(): UNKNOWN INTERFACE (GS3DERR 188)
189	*** BUS ERROR OCCURRED BEFORE EXECUTING TEST LOOP. (GS3DERR 189)
190	*** BUS ERROR NOT CAUGHT BY SECTION; NOW ABORTING SECTION. (GS3DERR 190)
CAUSE:	These error messages should only be reported on series 300 or during series 800 factory use.
EFFECT:	If this error occurs during online diagnostic operation, please notify the factory
191	*** FLUSH_STAT_LOG(): NON-ZERO <== FFLUSH() (GS3DERR 191)
192	*** FLUSH_ERROR_LOG(): NON-ZERO <== FFLUSH() (GS3DERR 192)
193	*** CANNOT OPEN PICTURE FILE "!". (GS3DERR 193)
194	*** ERROR. ANIM(): READ() SYSTEM CALL RETURNED ERROR. (GS3DERR 194)
195	*** ERROR: TIMED OUT WHILE WAITING FOR RESOURCE '!' TO UNLOCK. (GS3DERR 195)
196	*** ERROR: TIMED OUT WHILE WAITING FOR CHAMBER TO POWER UP. NOTE: TYPICALLY CRONTAB INVOKES THE PROCESS THAT READS THE CHAMBER CONTROLLER AND WRITES THE RESULT INTO CHAMBER.STATUS IN THE UCODE_DIRECTORY. IF CRONTAB DIES AND CHAMBER.STATUS READS POWER OFF, THEN A TIMEOUT WILL OCCUR. (GS3DERR 196)
197	*** GOOD LUCK ! (GS3DERR 197)

198	*** ****************************** * BUS ERROR ENCOUNTERED WHILE WRITING FRAME BUFFER ENABLE REGISTER ON* * INTERFACE CARD. UNABLE TO LOCK FRAME BUFFER SPACE. PLEASE POWER DOWN THIS * * DAVINCI AND ENTER A CARRIAGE RETURN. FAILURE TO DO SO WILL CAUSE OTHER * *TESTS RUNNING ON THIS SPU TO FAIL * ******************************** (GS3DERR 198)
199	*** ****************************** * BUS ERROR ENCOUNTERED WHILE WRITING FRAME BUFFER ENABLE REGISTER ON * * INTERFACE CARD. UNABLE TO UNLOCK FRAME BUFFER SPACE. PLEASE POWER DOWN * * THIS DAVINCI AND ENTER A CARRIAGE RETURN. FAILURE TO DO SO WILL CAUSE * * OTHER TESTS RUNNING ON THIS SPU TO FAIL. * ****************************** (GS3DERR 199)
200	*** ERROR ARM_SIGNAL_TRAPPING(): -1 <== SIGVECTOR() (GS3DERR 200)
201	*** PROCESS_FAIL_FLAG(): UNKNOWN TEST RESULT VALUE. (GS3DERR 201)
202	*** PRINT_STATISTICS(): UNKNOWN TEST RESULT VALUE. (GS3DERR 202)
203	*** OBJ_DLOAD(): MASTER BOARD ABSENT (GS3DERR 203)
204	*** OBJ_DLOAD(): MASTER OR SLAVE BOARD ABSENT (GS3DERR 204)
205	*** OBJ_DLOAD(): UNSUPPORTED ENGINE SELECTOR '!' (GS3DERR 205)
206	*** DOWNLOAD SCRIPT '!' MISSING. (GS3DERR 206)
207	*** INVALID INTERFACE CONFIGURATION NUMBER. FORMAT: 0X00000S S = I/O CARD SWITCH SETTING = 2..8 (GS3DERR 207)

208	*** INVALID INTERFACE CONFIGURATION NUMBER. FORMAT: 0X000SSF DIO I SPACE: (VOLVO I, VOLVO II) SS = CONTROL SPACE SWITCH SETTING = 00..1F F = FRAME BUFFER LOCATION = 2 OR 8 DIO II SPACE: (VOLVO II) SS = CONTROL SPACE SWITCH SETTING = 01..1F F = 4. FRAME LOCATION DERIVED FROM SS (GS3DERR 208)
209	*** CHECK YOUR DEVICE FILES, INTERFACE AND INTERFACE CONFIG. (GS3DERR 209)
210	*** ERROR: CAN'T OPENED FRAME BUFFER DEVICE FILE: "!" (GS3DERR 210)
211	*** CAN'T OPENED LOWER FRAME BUFFER DEVICE FILE: "!" (GS3DERR 211)
212	*** CAN'T OPENED UPPER FRAME BUFFER DEVICE FILE: "!" (GS3DERR 212)
213	*** ERROR: CAN'T OPEN CONTROL SPACE DEVICE FILE: '!' (GS3DERR 213)
214	*** CAN'T OPENED FRAME/CONTROL/HPA SPACE DEVICE FILE: "!" (GS3DERR 214) NOTE: SPU MUST BE POWERED UP WITH THE DAVINCI CONNECTED IN ORDER TO OPEN THE DEVICE FILE.
215	*** CAN'T IOMAP FRAME BUFFER. (GS3DERR 215)
216	*** CAN'T IOMAP LOWER FRAME BUFFER. (GS3DERR 216)
217	*** CAN'T IOMAP UPPER FRAME BUFFER. (GS3DERR 217)
218	*** CAN'T IOMAP CONTROL SPACE. (GS3DERR 218)
219	*** CAN'T GCDESCRIBE FRAME/CONTROL/HPA SPACE. (GS3DERR 219)

220	*** CAN'T GCMAP FRAME/CONTROL/HPA SPACE. (GS3DERR 220)
221	*** ERROR. MEMORY_MAP_SPECTROGRAPH(). CRT_ID IS NOT A DAVINCI DEVICE. (GS3DERR 221)
222	*** ERROR. MEMORY_MAP_SPECTROGRAPH(): UNSUPPORTED S300 INTERFACE. (GS3DERR 222)
223	*** ERROR. MEMORY_MAP(): UNSUPPORTED TESTCODE CUSTOMER. (GS3DERR 223)
224	*** ERROR. INVALID INTERFACE CONFIGURATION NUMBER. (PARM1). (GS3DERR 224)
225	*** ERROR. MISSING INTERFACE CONFIGURATION NUMBER. (PARM1). (GS3DERR 225)
226	*** ERROR. NIMBUS INTERFACE IS UNSUPPORTED ON THIS SYSTEM. (PARM2). (GS3DERR 226)
227	*** ERROR. MEMORY_MAP(): INVALID INTERFACE. ENUMERATED VALUE = !. (GS3DERR 227)
228	*** ERROR. UNSUPPORTED DAVINCI TEST COMMAND NAME. (GS3DERR 228)
229	*** ERROR. UNSPECIFIED INTERFACE CARD. (PARM2) (GS3DERR 229)
230	*** ERROR. INVALID CONFIGURATION ARGUMENT. (PARM5) (GS3DERR 230)
231	*** ERROR. INVALID LOOP_COUNT ARGUMENT. (PARM7) (GS3DERR 231)
232	*** INVALID CONFIGURATION COMMAND LINE ARGUMENT. (GS3DERR 232)
233	*** INVALID INTERFACE COMMAND LINE ARGUMENT. "!" (GS3DERR 233)
234	*** ERRNO = EINVAL (ADDR OUT OF RANGE OR IOCTL CMD INVALID) (GS3DERR 234)
235	*** ERRNO = ENOMEM (CANNOT ALLOCATE REQUIRED MEMORY) (GS3DERR 235)

236	*** ERRNO = ENODEV (READ/WRITE UNSUPPORTED) (GS3DERR 236)
237	*** ERRNO = ENXIO (NO SUCH ADDRESS) (GS3DERR 237)
238	*** ERRNO = ENOSPC (CANNOT ALLOCATE REQUIRED RESOURCES) (GS3DERR 238)
239	*** ERRNO = ENOTTY (BAD IOCTL CMD OR IOCTL OF OPEN FILE) (GS3DERR 239)
240	*** ERRNO = EAGAIN (OPERATION WOULD SUSPEND PROCESS) (GS3DERR 240)
241	*** ERRNO = EBUSY (DEVICE ALREADY LOCKED DURING GCLOCK) (GS3DERR 241)
242	*** ERRNO = EINTR (IOCTL() CALL WAS INTERRUPTED) (GS3DERR 242)
243	*** ERRNO = EPERM (DEVICE ALREADY LOCKED DURING GCUNLOCK) (GS3DERR 243)
244	*** VTAF.C PROCESS_TEST(): TEST SEQUENCING ERROR. (GS3DERR 244)
245	*** WARNING: NO TEST DISABLE FILE ARG. ALL TESTS ENABLED. (GS3DERR 245)
246	*** ERROR OPENING TEST DISABLE FILE: "!" DEFAULT ACTION. ALL TESTS ENABLED. (GS3DERR 246)
247	*** ERROR: UNKNOWN TEST NAME: "!" IN FILE "!" (GS3DERR 247)
248	*** PROCESS_TEST(): NUMBER OF OPEN FILES IS INCREASING. (GS3DERR 248)
249	*** PROCESS_TEST(): FRAME BUFFER LOCKED FOLLOWING TEST. (GS3DERR 249)
250	*** ERROR: TOO MANY ERRORS. TEST ABORTED. (GS3DERR 250)
251	*** ERROR: PROCESS_TEST_RESULT(): UNKNOWN BOOLEAN RESULT RETURNED FROM TEST (GS3DERR 251)
252	*** ERROR: TEST_LOOP(): TEST INVOKED WITH UNSUPPORTED NAME. (GS3DERR 252)

253	*** ERROR: NULL <== FOPEN(STAT_FILE). CHECK PERMISSIONS AND FILE SPACE SIZE. (GS3DERR 253)
254	*** ERROR: NULL <== FOPEN(DATA_BASE). CHECK PERMISSIONS AND FILE SPACE SIZE. (GS3DERR 254)
255	*** ERROR: NULL <== FOPEN(DIAG_FILE). CHECK PERMISSIONS AND FILE SPACE SIZE. (GS3DERR 255)
256	*** VTAF.C MAIN(): BUS ERROR TEST ESCAPE IN CURRENT MODULE. (GS3DERR 256)
257	*** CONTROL SPACE DEVICE FILE CANNOT BE LOCKED. LOCKED BY ANOTHER TEST PROCESS. (GS3DERR 257)

7

98720A Graphics Processor Diagnostic

Introduction

The HP 98720A Graphics Processor Diagnostic (GP3DDIAG) provides an online test of 98720A Graphics Processor(s) on any HP Precision Architecture RISC computer system which supports the Online Diagnostic subsystem. The Processor can be configured as the system console or a graphics workstation. The type of tests that will be provided can verify all functions of the 98720A Graphics Processor.

When a test function associated with a particular board has failed, replace that (FRU).

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10015.

Minimum Configuration

GP3DDIAG is designed to test and verify 98720A Graphics Processors configured as system consoles or workstations on any PA-RISC computer. The system must be up and running on HP-UX and have a A1017A board correctly installed and configured in the 98720A.

Operating Instructions

The 98720A Graphics Processor Diagnostic is accessed by the user via the Diagnostic User Interface. It is initiated using the **run gp3ddiag** command. Refer to the DUI section for details concerning the **run** command and its parameters.

Because all sections are destructive, the diagnostic requires that all sections be run in Single User Mode (SUM). Only those users with level 0 security will be able to execute all sections.

Default Tests

If the user does not specify which sections to run, all of the sections will be run.

RUN Command

All available **run** command parameters are acceptable when running the diagnostic.

To invoke the On-Line Diagnostics Subsystem, enter the following at the system prompt:

```
%sysdiag
```

Enter the following command at the DUI prompt:

```
DUI>run gp3ddiag pdev=8 <RUN Command Options>
                         |            |
                         | none required for
                         | default test suite
                         |
                    insert physical location of
                    device to be tested here;
                    or type the devfile name
```

When GP3DDIAG is run, a header and a welcome message will be displayed.

At this point, the diagnostic prompts the user with the following four questions:

```
Do you have a SCAN BOARD ([y]/n) ?

Do you have a TRANSFORM BOARD ([y]/n) ?

How many FRAME BUFFER BOARDS do you have (0-[4]) ?

Are you running GP3DDIAG from the 98720A you are testing ([y]/n) ?

Enter the type of Series 800 interface card you are using.
1= A1017A (SPECTROGRAPH)            2 = A1066A (LEONARDO)
```

Note

If the user runs GP3DDIAG from the graphics subsystem as the system console, CTRL -C should be pressed to interrupt the diagnostic: the CTRL SHIFT RESET keys should also be pressed to reset the graphics subsystem, and to allow GP3DDIAG to return to the DUI> prompt.

After answering the questions, the sections specified by the user will be executed and the results output. If the user did not specify sections to be run, all sections will be run.

Note

When you are prompted for the number of frame buffer boards present, you must specify the exact number, because GP3DDIAG expects to find an empty slot for nonexistent frame buffer boards. If you do not specify the exact number, the diagnostic will report a CRC error against the boards configured.

If at any time the number of errors generated reaches the limit specified by the user (10 for instance) in the ERRCOUNT parameter of the **run** command, the following message will be output:

```
Error count exceeded. Count = 10
```

The diagnostic will then terminate execution. If the TRACE parameter is used, and equated to PROGRAMS or ALL, then extended informational messages are displayed along with the diagnostic output. If the diagnostic is run from the 98720A which is being tested, the TRACE info should be sent to a file to avoid corrupting visual output.

If the ERRPAUSE parameter of the **run** command was assigned a value of "on", then this diagnostic will stop after each error is generated and ask the user if testing should continue:

```
Error pause -- do you wish to continue ([y]/n) ?
```

If the response is "Y", then the test will be resumed (if possible). If the response is "N", then the diagnostic terminates. The "[Y]" indicates that "Y" is the default response if the user simply hits Return in response to the prompt. If the sections and steps specified by the user were executed the number of times specified in the LOOP parameter of the **run** command without the number of errors exceeding the ERRNUM value, the diagnostic terminates normally. Upon termination of this diagnostic, control returns to the diagnostic system.

Test Section Descriptions

There are 31 test sections in GP3DDIAG.

Section 10	Allegro Monitor
Section 11	Refresh Bus
Section 12	Z-Buffer
Section 13	Repeat Pattern
Section 14	Dither
Section 15	Transparency
Section 16	Frame Buffer RAM via LGB
Section 17	Transform Board Registers
Section 18	Simple Test WCS
Section 19	IEEE Writeable Control Store Memory
Section 20	Writeable Control Store Memory Walking Bit
Section 21	Transform Board Sequencer
Section 22	Transform Board ALU
Section 23	Transform Board Pointer RAM
Section 24	Transform Board Data RAM
Section 25	Transform Board Floating Point Chip
Section 26	Command Data RAM Path
Section 27	DC RAM via LGB
Section 28	DC RAM via uCode
Section 29	ACE Register
Section 30	Color Map
Section 31	ID Font/ROM
Section 32	Frame Buffer Controller Shadow RAM
Section 33	Frame Buffer RAM
Section 34	Frame Buffer Controller Write Enable
Section 35	Frame Buffer Controller Folded/Normal Mode Addressing
Section 36	Frame Buffer Controller Window Move
Section 37	Frame Buffer Controller Slow Window Move
Section 38	ACE Chip
Section 39	Real Time Measurements
Section 40	Transform Board Spin

Each section contains an explanation which consists of the actions performed. and the expected normal output if no errors occur. Visual output may vary depending on the 98784A configuration and the sequence of tests run. Some tests may flash the screen if the system is under heavy usage. For a complete list of error/information messages that may be generated while running GP3DDIAG, please refer to the Information/Error subsection at the end of this diagnostic.

Section 10—Allegro Monitor

This section verifies the 98784A monitor and color map functionality. The test is only visual; no real error detection is performed by the program. This test requires that block moves work. The user will see these colors:

- White full screen
- Red full screen
- Green full screen
- Blue full screen
- Four concentric circles and a grid. The inner 3 circles have radii of 1, 2, and 3 inches. The outer circle touches the resolution boundaries. All patterns should be white.
- Convergence pattern—an array of #'s.
- Color Bars. There should be two repetitions of vertical color bars. The colors are (from the left):
  - Black
  - Blue
  - Green
  - Greenish-Blue
  - Red
  - Pink-Red
  - Yellow
  - White

All these patterns are produced in the overlay planes and should be independent of the number of frame buffer planes. On a single loop the user will have to (Return) between colors to get the next one. If "Loop" > 1 only the first loop will ask for (Return)'s.

Output:

Visual only. The patterns are listed above.

Section 11—Refresh Bus

This section displays patterns which are useful in finding refresh bus problems. The test is only visual; no real error detection is performed by the program.

This test requires that block moves and fast screen clears work. This test generates 256 bars with frame buffer values (color map indexes) of 0 to 255. Then the bits of the refresh bus (the path from the frame buffer to the color map) are enabled one at a time. This results in the various patterns that are seen on the screen (a series of pictures might be useful). The user should look for any flickering or unusual colors in the bar patterns. This would indicate a problem on the associated bit of the refresh bus. The first bit enabled is the LSB.

Output:

Various bar patterns in gray or white.

Section 12—Z-Buffer

This section verifies the Z-Buffer functionality. It requires that fast screen clears and polygons work. This test verifies the Z-buffer using the intersection of two quadrilaterals. All boards present are used for the z-buffer. The z-buffer is divided up on 64 pixel boundaries and exists only in the non-displayed areas of the frame buffer. This section also verifies that z-buffering can be turned off. In addition the ZKEEP. ZEQ and ZPASS circuitry is tested.

Output:

The screen should have three rows of intersecting quadrilaterals. They will initially be gray but turn to blue/pink when the Cyclic Redundancy Checking (CRC) is being performed. The first row will vary in length depending on how many frame buffer boards are installed. The second row has the test patterns for z-buffer off, ZEQ and ZKEEP. There should be four intersecting quadrilaterals. The last row should contain one rectangle with a bright right edge. This is used to test the ZPASS circuitry. The colors may change depending upon the number of frame buffer planes present in the system.

Section 13—Repeat Pattern

This section verifies repeat patterning. All repeat lengths in X and Y are tested individually although only a few combinations can be tested because of the large number present.

This test requires that fast screen clears and block moves work correctly. First, a pattern is drawn at 1024,0 (the square with the "zebra" pattern). Then rectangles are drawn using this pattern. The rectangles are drawn large enough to cycle through the pattern at least once. The repeat widths tested are: 16. 32. 64. 128 and 256 (all with height of 4). The repeat heights tested are 8, 16. 32. 64. 128 and 256 (all with width of 256). The result is checked with a CRC.

Output:

The screen should have a "zebra" pattern in white. The color may change when the CRC begins depending on the number of frame buffer planes.

Section 14—Dither

This section checks the three modes of dither and requires that fast block moves work. The test section verifies three modes of dither (3:3:2, 4:4 and indexed) by drawing three large quadrilaterals with these modes and doing a CRC. The upper quadrilateral tests 3:3:2 mode dither, the middle tests indexed and the bottom tests 4:4 mode dither.

Output:

There should be three large right-canted quadrilaterals on the screen. The upper one should fade to black towards its center. The middle one should have fringes of white in the upper left and lower right corners. The bottom one should have fringes of colors in all corners. The colors may change depending on the number of frame buffer boards present. They may also change while the CRC is being performed.

Section 15—Transparency

This section verifies the transparency feature and requires that fast screen clears work. The following transparency values are checked: 5a5a, a5a5, 0f0f, f0f0, 5555 and cccc. This assures that all bits can be toggled on and off and that adjacent bits don't interact. The results are checked with a CRC.

Output:

There should be 6 squares slightly to the left of the center of the screen. They may change in color depending upon the number of frame buffer planes installed. Careful inspection will show various patterns in the fills of the squares.

Section 16—Frame Buffer RAM via LGB

This section verifies frame buffer RAM memory board functioinality including overlay planes directly via the SPU via the LGB. No ucode is required.

This section will not run if a Transform Board is installed. The memory test algorithm used for the IEEE_WCS test is implemented by the SPU and applied to the frame buffer memory. The test is applied to each board serially by operating the boards in folded mode and walking an enable bit through the fbdrive register. The color map is loaded so that each board displays a unique color:

```
BdO:     Blue
Bd1:     Green
Bd2:     Red
Bd3:     (Non-Displayable. Only used for Z-Buffer)

Overlay Planes: Invisible unless the BdO is absent.  In this case the hardware
                automatically maps the overlay into BdO address space and it is
                displayed blue.
```

Boards 0..3 display identical outputs except that the color is different. The display associated with bd0 is outlined below.

Output:

```
- Clear screen
- Dark  blue screen entirely filled from top to bottom.
- Clear entire screen from top to bottom.
- Dark  blue screen entirely filled from bottom to top.
- Clear entire screen from bottom to top.
- Dark  blue screen entirely filled from top to bottom.
- Clear entire screen from bottom to top.
```

Section 17—Transform Board Registers

This section verifies functionality of the Transform Board HALT, STATUS and ID registers. The HALT and STATUS registers must be functional for any other test which uses the Transform Engine to pass. (All CRC operations use the Transform Engine)

```
Read Transform I.D. Register. Expects to see OxFFF8.

Starts and stops Transform Engine and reads back the Transform status
register to verify operation.
```

Output:

Error message No visual output.

Section 18—Simple Test WCS (Random Control Store Test)

This section verifies the control store memory on the Transform Board by writing a sequence of pseudo random values into memory followed by a verification sequence. This routine finds pattern sensitivities that may go undetected with the TEST_WCS_MEMORY test.

```
Write entire control store sequentially with random values.
Read and compare control store with values written.
```

Output:

Error message No visual output.

Section 19—IEEE Writeable Control Store Memory

This section verifies the control store memory on the Transform Board. Control store memory must be functional for any test to pass which depends on Microcode (ucode).

All tests using crc's require ucode. The IEEE algorithm used operates with single bit word size RAMs. It is modified for multi-bit word arrays by assuming no interaction within the RAM arrays between bits of a word. If this is the case, then the algorithm is extended by writing, reading and testing with data values of zero and -1 only. Since the assumption of independence of RAM bits cannot be guaranteed, the testing is further extended in the WRITEABLE CS MEMORY WALKING BIT TEST.

Output:

Error messages only. No visual output.

Section 20—Writeable Control Store Memory Walking Bit

This section performs a walking bit test on a single micro address of the writeable control store (WCS) memory. The purpose of this test is to check for shorts on the memory busses which go undetected by the WCS IEEE based test. (i.e., WCS memory reads and writes only 0 and -1). To provide sufficient fault coverage the bit must be walked across every RAM in the system at a single address. Since the RAMs are not stacked to increase depth, (16k x 4 bit RAMs - 18 RAMs used) only a single micro address is required. This test must be functional for other test to pass which depend on ucode (All CRC based tests).

Output:

Error messages only. No visual output.

Section 21—Transform Board Sequencer

This section verifies the Sequencer functionality which is part of the Transform Engine. A special ucode routine is downloaded to test the sequencer. If the Transform Engine is functional enough to transmit results back to the SPU, a detailed explanation of the failure mechanism is output in an error message.

The results are useful for sub-board diagnostics. The hardware is reset (Also Halts Transform Engine), and an appropriate ucode file is downloaded. At this point the Transform Engine is started by writing to the halt register. After the IBF flag is set, the input register on the Transform Board is read. If most significant bit (MSB) is set, an error has occurred and the 15 least significant bits (LSBs) create an error number which is converted into a failure description.

Output:

Error messages only. No visual output.

Section 22—Transform Board ALU

This section verifies Arithmetic Logic Unit (ALU) functionality which is part of the Transform Engine. A special ucode routine is downloaded to test the ALU. If the Transform Engine is functional enough to transmit results back to the SPU, a detailed explanation of the failure mechanism is output in an error message.

The results are useful for sub-board diagnostics. The hardware is reset (Also Halts Transform Engine), and an appropriate ucode file is downloaded. At this point the Transform Engine is started by writing to the halt register. After the IBF flag is set, the input register on the Transform Board is read. If MSB is set, an error has occurred and the 15 LSBs are an error number which is converted into a failure description.

Output:

Error messages only. No visual output.

Section 23—Transform Board Pointer RAM

This section verifies Pointer RAM functionality which is part of the Transform Engine. A special ucode routine is downloaded to test the Pointer RAM. If the Transform Engine is functional enough to transmit results back to the SPU, a detailed explanation of the failure mechanism is output in an error message.

The results are useful for sub-board diagnostics. The hardware is reset (Also Halts Transform Engine), and an appropriate ucode file is downloaded. At this point the Transform Engine is started by writing to the halt register. After the IBF flag is set, the input register on the Transform Board is read. If MSB is set, an error has occurred and the 15 LSBs create an error number which is converted into a failure description.

Output:

Error messages only. No visual output.

Section 24—Transform Board Data RAM

This section verifies Data RAM functionality which is part of the Transform Engine. A special ucode routine is downloaded to test the Data RAM. If the Transform Engine is functional enough to transmit results back to the SPU, a detailed explanation of the failure mechanism is output in an error message.

The results are useful for sub-board diagnostics. The hardware is reset (Also Halts Transform Engine), and an appropriate ucode file is downloaded. At this point the Transform Engine is started by writing to the halt register. After the IBF flag is set, the input register on the Transform Board is read. If MSB is set, an error has occurred and the 15 LSBs create an error number which is converted into a failure description.

Output:

Error messages only. No visual output.

Section 25—Transform Board Floating Point Chip

This section verifies Floating Point Chip functionality which is part of the Transform Engine. A special ucode routine is downloaded to test the Floating Point Chip. If the Transform Engine is functional enough to transmit results back to the SPU, a detailed explanation of the failure mechanism is output in an error message.

The results are useful for sub-board diagnostics. The hardware is reset (Also Halts Transform Engine), and an appropriate ucode file is downloaded. At this point the Transform Engine is started by writing to the halt register, after the IBF flag is set, the input register on the Transform Board is read. If MSB is set, an error has occurred and the 15 LSBs are an error number which is converted into a failure description.

Output:

Error messages only. No visual output.

Section 26—Command Data RAM Path

This section verifies the path to the command data RAM on the Transform Engine board. The command data RAM can only be written (not read) from the LGB. This tests the path from the LGB to the command data RAM by writing various patterns into the RAM and using ucode (downloaded into the Transform Engine) to read back the results.

Output:

Error messages only. No visual output.

Section 27—DC RAM via LGB

This section verifies the DC RAM memory through the LGB port. The device coordinate RAM (DC RAM) resides on the scan board and is used to store commands and intermediate results for the ACE chip (Scan Converter). There are three ports into this memory, Ace, LGB and the Transform Engine. This section performs the IEEE memory test on the DC RAM similarly to the IEEE Writeable Control Store Memory' test. In addition, a walking bit test is performed as in the 'Writeable Control Store Walking Bit' test. All communication with DC RAM is via direct LGB reads and writes to the scan board.

Output:

Error messages only. No visual output.

Section 28—DC RAM via uCode

This section verifies the DC RAM memory through the Transform Engine port using ucode. The device coordinate RAM (DC RAM) resides on the scan board and is used to store commands and intermediate results for the ACE chip (Scan Converter). There are three ports into this memory, Ace, LGB and the Transform Engine. This section uses ucode to write patterns to the DC RAM and also uses ucode to test the results. If an error is found the ucode flags the SPU. The ucode passes the error information to the SPU and it is put into an error message.

Output:

Error messages only. No visual output.

Section 29—ACE Register

This section isolates Ace chip (resident on scan board) faults by testing a significant portion of the chip while relying on minimum hardware. The section performs a walking bit test on all Ace registers simultaneously. To do this a virtual register is constructed containing all Ace registers. All single one bit and single zero bit patterns are shifted through the virtual register. To perform the necessary reads and writes to Ace. DC memory is continually loaded with Ace commands. The register address and contents are listed below:

Address	Contents
0	Pointer to Read or Write Routines. (2 or 5)
1	Halt Instruction
2	Write Register Instruction
3	Write Immediate Data
4	Halt Instruction
5	Read Instruction
6	Indirect Address
7	Halt Instruction
8..25	Result of Read Instructions
26	Halt Instruction

Output:

Error messages only. No visual output.

Section 30—Color Map

This section verifies the colormap status register and colormap memory. The status register is read and various bits tested.

Output:

Error messages only. No visual output.

Section 31—ID Font/ROM

This section verifies that the ID/Font ROM (located on the frame buffer controller) contains valid information. and is readable via the LGB. It also checks the primary device ID contained in the ID ROM on the frame buffer controller board. Should be 0x39 and checks the secondary device ID contained in the ID ROM on the frame buffer controller board. Should be 0x04. Then the section reports the current ROM revision but does not test to insure a particular revision is loaded. Finally, it performs a check sum of the ROM. The roms are programmed so the check sum is 0xffff independent of revision.

Output:

Error messages only. No visual output.

Section 32—Frame Buffer Controller Shadow RAM

This section verifies shadow RAM functionality but not necessarily the corresponding control storage elements. Many of the control registers located on the frame buffer controller board operate with elements that are not readable by the SPU.

To support the operating system requirement that all control registers be readable, the concept of shadow RAM was created. When a real control element is written a RAM location is written simultaneously. When the SPU reads the same address the contents of the RAM is returned to the SPU. An IEEE test similar to the IEEE_WCS test is performed including the walking bit test.

Output:

Error messages only. No visual output.

Section 33—Frame Buffer RAM

This section verifies the frame buffer RAM memory boards including overlay planes. The memory test algorithm used for the IEEE_WCS test is implemented in ucode to decrease test time. This test is applied to each board serially by operating the boards in folded mode and walking an enable bit through the fbdrive register.

The color map is loaded so that each board displays a unique color.

```
Bd0:     Blue
Bd1:     Green
Bd2:     Red
Bd3:     (Non-Displayable. Only used for Z-Buffer)

Overlay Planes: Invisible unless the Bd0 is absent.  In this case the
                hardware automatically maps the overlay into Bd0 address
                space and it is displayed blue.

If an error is found the ucode flags the SPU.  The ucode passes the error
information to the SPU and it is put into the error log.
```

Bds 0..3 display identical outputs except that the color is different. The display associated with bd0 is outlined below.

Output:

```
- Clear screen
- Dark  blue screen entirely filled from top to bottom.
- Light blue screen entirely filled from top to bottom.
- Dark  blue screen entirely filled from top to bottom.
- Clear entire screen from top to bottom.
- Dark  blue screen entirely filled from bottom to top.
- Light blue screen entirely filled from bottom to top.
- Dark  blue screen entirely filled from bottom to top.
- Clear entire screen from bottom to top.
- Light blue screen entirely filled from bottom to top.
- Dark  blue screen entirely filled from top to bottom.
```

Section 34—Frame Buffer Controller Write Enable

This section verifies the WRITE ENABLE and DRIVE register electronics functionality located on the frame buffer controller board. Certain address manipulator functions are tested by writing in folded mode and reading back in full mode. All frame buffer accesses are direct LGB reads and writes.

For 300s (DIO 1 Only), the test code is capable of testing up to 4 graphic processors simultaneously on a single SPU. This is accomplished by installing 4 interface cards into the DIO backplane and configuring each card to operate out of a unique control space. Unfortunately, DIO 1 does not provide enough memory space to map each frame buffer uniquely. Therefore, all 4 frame buffer memories reside in the same space.

A register located in the interface card selects which graphic processor frame buffer is active. An IPC semaphore structure is setup to coordinate this activity among the 4 test processes. If the interface card should fail while multiple graphic processors are being tested, then all 4 tests may report an error.

Walking Write Enable Bit Test (Full Mode. Drive = $0f): This walks a bit through all 32 bits of the FBWEN and OPWEN registers with the DRIVE register contents equal to 0x0d. (Overlay planes disabled). At each value of WEN, write -1 and 0 into the frame buffer and verify that the appropriate planes have been written to.

The FBWEN register is a 32 bit register that selects which of the possible 32 planes on the 4 frame buffer boards are enabled for writing. The OPWEN register is an 8 bit register which selects which of the possible 4 overlay planes are enabled to be written. The overlay planes reside in the same space as frame buffer board 0. In this test, the OPWEN register is written with the least significant 8 bits of the value written into FBWEN.

Walking Write Enable Bit Test (Full Mode. Drive = $1e): The process is repeated with DRIVE register equal to 0x1e. (Overlay planes enabled and frame buffer board 0 disabled)

Walking Drive Bit Test (Folded Mode): In this test all frame buffer boards are written at a single address in folded mode with the same data simultaneously by setting the drive register to $1f. The boards are individually read back and verified by walking an enable bit though the DRIVE register. The process is repeated with unique data written into each frame buffer board.

Walking Drive Bit Test (Full Mode): The frame buffer controller is configured for full mode and the data written in the test above is read and verified.

Output:

Error messages only. No visual output.

Section 35—Frame Buffer Controller Folded/Normal Mode Addressing

This section verifies the address manipulator functionality located on the Frame Buffer Controller and tests out some of the same hardware as 'FRAME BUFFER WE Test'. As in the 'FRAME BUFFER WE Test' the frame buffer space must be locked. In this test an area of the frame buffer is written in folded mode and read back in full (normal) mode. Likewise, the same area is written in full mode and read back in folded mode. The space tested is a 10 line by 40 pixel rectangular area at the upper left of the frame buffer and also the lower right.

Output:

Error messages only and no visual output.

Section 36—Frame Buffer Controller Fast Window Move

This section verifies block mover circuit functionality located on the frame buffer controller. The verification is performed while functioning in 128 bit wide pixel operations. The block mover operating in fast block move mode is checked. In this mode the operations are restricted to simple operations such as write source with a granularity of 128 pixels. Under these conditions the block mover is very fast. Several special cases of the mover are used to create a pattern on the display. A CRC is taken of the frame buffer using the Transform Engine and special ucode and is then compared to an expected CRC.

Output:

Error messages and visual. An image resembling the one below is created on the screen constructed of 128 bit wide lines. The color of the image is a function of the number of boards loaded. Bd0 only appears as blue, Bd0 and Bd1 appears as magenta and Bd0..2 appear as white. Bd3 and the overlay planes has no effect on color.

```
 | |   ---
-----
 | |   ---
 | |   ---
```

Section 37—Frame Buffer Controller Slow Window Move

This section verifies block mover circuit functionality located on the frame buffer controller, while functioning in single pixel resolution mode. All block mover replacement rules and frame buffer location permutations of source, pattern and destination are exercised to create the unusual image depicted below on the screen. A CRC is taken of the frame buffer using the Transform Engine and special ucode and is then compared to an expected CRC to ensure that the results are essentially the same.

Output:

Error messages and visual. An image resembling the image below is created on the screen constructed mostly of sparsely filled rectangles. The rectangles appear to be filled with a random sequence of dots. The sequence is not random and every rectangle is different. The color of the image is a function of the number of boards loaded. Bd0 only appears as blue, Bd0 and Bd1 appears as magenta and Bd0..2 appear as white. Bd3 and the overlay planes have no effect on color.

```
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |


| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
      | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
          | | | | | | | | | | | | | | | | | | | | | | | | | | |

          |         |         |
          |         |         |
```

Section 38—ACE Chip

This section verifies ACE chip functionality. The section steps are as follows.

- The first draws a full screen rectangle which utilizes all six interpolators.

The second tests the linetype ability and also draws some polygons

- which test swapping data in and out of RAM.
- The third test draws two triangles which test the ACE output formatter.

Output:

The first display is a multicolor full screen rectangle which should be white in the upper left hand corner, red in the upper right hand corner, green in the lower left hand corner and blue in the lower right hand corner. These colors may change if less than the full complement of planes is present.

The second display consists of quite a few horizontal and nearly horizontal lines. In addition there are some small polygons of various shapes.

The third test displays 2 triangles - one with red, green and blue vertices and one with a blue apex. CRC's are done on each of the three displays to determine if there has been an error.

Section 39—Real Time Measurements

This section verifies the timing of various signals or processes in the 98720A. All timing tests use the same algorithm. A ucode program monitors conditionally for a branch which will start it to count. This program continues counting until a change signal condition causes the count halt. The ucode passes the number of counts to the SPU and it determines if the count is correct.

The following signals are tested:

Color map status register blink bit

Color map status register not copy bit Color map status register H blank bit Color map status register V sync bit Fbufc Vblank bit Fbufc Intv Wbusy fast move Intb fast move Wbusy slow move source Intb slow move source Wbusy slow move destination Intb slow move destination Wbusy slow move pattern Intb slow move pattern Acebusy

Output:

Error messages only. No visual output.

Section 40—Transform Board Spin

This section verifies some otherwise untested areas of the Transform Board. The test exercises the Transform Board in a manner similar to its use by the customer. Microcode is used to exercise the various blocks of the board concurrently. The results are checked by taking CRC's of frame buffer RAM.

Output:

Error messages and visual. Two triangles, one stationary and one spinning inside the other will be seen.

Information Messages

The following is a list of information messages which may appear when using GP3DDIAG. Error messages are listed in the next section. Other diagnostic error messages may appear at any time. Error messages without the (GPERR #) or (GPMSG #) trailer are generated by the Online Diagnostic subsystem or the Operating system. The "!" indicates that a parameter of some sort replaces the exclamation point when the message is displayed.

2006	FAST WINDOW MOVE (!, !) W = ! H = ! RULE = !. (GPMSG 2006)
2007	RESETTING HARDWARE. (GPMSG 2007)
2008	TEST SKIPPED: TRANSFORM BOARD REQUIRED. (GPMSG 2008)
2009	TEST SKIPPED: SCAN BOARD REQUIRED. (GPMSG 2009)
2010	TEST SKIPPED: AT LEAST ONE FRAME BUFFER BOARD REQUIRED. (GPMSG 2010)
2011	TRANSFORM BOARD PRESENT. (GPMSG 2011)
2012	SCAN BOARD 0 ABSENT. (GPMSG 2012)
2013	SCAN BOARD PRESENT. (GPMSG 2013)
2014	FRAME BUFFER BOARD ! PRESENT. (GPMSG 2014)
2015	FRAME BUFFER BOARD ! ABSENT. (GPMSG 2015)

2016	OVERLAY PLANES ABSENT. (GPMSG 2016)
2017	OVERLAY PLANES PRESENT. (GPMSG 2017)
2106	WRITING COLORMAP RAM WITH !. (GPMSG 2106)
2205	DOING CRC FOR ACE TEST !. (GPMSG 2205)
2206	SETTING UP ZMAP. (GPMSG 2206)
2207	SETTING Z-BUFFER. (GPMSG 2207)
2208	TURNING Z-BUFFER OFF. (GPMSG 2208)
2209	TESTING Z KEEP. (GPMSG 2209)
2210	TESTING ZEQ - ZEQ ON. (GPMSG 2210)
2211	TESTING ZEQ - ZEQ OFF. (GPMSG 2211)

2212	TESTING ZPASS. (GPMSG 2212)
2215	CLEARING SCREEN. (GPMSG 2215)
2216	SETTING UP COLOR MAP. (GPMSG 2216)
2217	SETTING UP PATTERN. (GPMSG 2217)
2218	BLANK PLANES. (GPMSG 2218)
2219	SETTING FBC VARIABLES. (GPMSG 2219)
2229	INIT OVERLAY/COLOR MAP TO PRIMARY COLORS. (GPMSG 2229)
2234	INIT PRIMARY & SECONDARY NON-OVERLAY CMAP !. (GPMSG 2234)
2235	INIT PRIMARY & SECONDARY OVERLAY CMAP TO !. (GPMSG 2235)
2236	INIT PRIMARY & SECONDARY NON-OVERLAY CMAP TO ADDRESS. (GPMSG 2236)

2237	INIT PRIMARY & SECONDARY NON-OVERLAY CMAP TO TRANSPARENT (GPMSG 2237)
2238	INIT PRIMARY & SECONDARY OVERLAY CMAP TO ADDRESS. (GPMSG 2238)
2239	INIT PRIMARY & SECONDARY OVERLAY CMAP TO GREY. (GPMSG 2239)
2307	CLEARING FRAME BUFFERS AND OVERLAY PLANES. (GPMSG 2307)
2309	TESTING FRAME BUFFER BOARD !. (GPMSG 2309)
2313	TESTING OVERLAY PLANE. (GPMSG 2313)
2314	TESTING OVERLAY PLANES USED AS FRAME BUFFER 0'S 4 PLANES. (GPMSG 2314)
2316	TESTING FRAME BUFFER AND OVERLAY PLANE RAM. (GPMSG 2316)
2317	WRITING FRAME BUFFER RAM WITH !. (GPMSG 2317)
2319	STEP ! UP. (GPMSG 2319)

2320	STEP ! DOWN. (GPMSG 2320)
2321	TEST SKIPPED. ONLY DONE WHEN TRANSFORM BD IS MISSING. (GPMSG 2321)
2322	TESTING FRAME BUFFER RAM VIA LGB IN FOLDED MODE. (GPMSG 2322)
2323	TOP LEFT CORNER IS ADDRESS 0. (GPMSG 2323)
2324	BOTTOM RIGHT CORNER IS ADDRESS 0X1FFFFF. (GPMSG 2324)
2401	INITIALIZE F.B.C. REGISTERS. (GPMSG 2401)
2404	BOARD !. (GPMSG 2404)
2407	OVERLAY. (GPMSG 2407)
2408	PLANE !. (GPMSG 2408)
2416	ALL PLANES. (GPMSG 2416)

2420	ROM REVISION !. (GPMSG 2420)
2424	WRITING F.B.C. SHADOW RAM WITH !. (GPMSG 2424)
2425	SEQUENCE ! UP. (GPMSG 2425)
2431	SEQUENCE ! DOWN. (GPMSG 2431)
2437	WALKING BIT (NORMAL SENSE). (GPMSG 2437)
2438	WALKING BIT (COMPLIMENT SENSE). (GPMSG 2438)
2440	FRAME_DEVICE_FILE_OPENED. (GPMSG 2440)
2441	WRITING FRAME BUFFER IN FULL MODE FROM LGB. (GPMSG 2441)
2442	READING FRAME BUFFER IN FOLDED MODE FROM LGB. (GPMSG 2442)
2444	WRITING FRAME BUFFER IN FOLDED MODE FROM LGB. (GPMSG 2444)

2445	READING FRAME BUFFER IN FULL MODE FROM LGB. (GPMSG 2445)
2518	SUCCESSFUL MAPPING OF POINTERS: = || FRAME BUFFER: ! = || CONTROL SPACE: ! = || (GPMSG 2518)
2519	UCODE VERIFY: UADDR = ! NUMBER_INT16 = !. (GPMSG 2519)
2522	ASSUMED CONFIGURATION. (GPMSG 2522)
2570	SYSTEM ROUTINE = READ (GPMSG 2570)
2571	SYSTEM ROUTINE = READV (GPMSG 2571)
2572	SYSTEM ROUTINE = WRITE (GPMSG 2572)
2573	SYSTEM ROUTINE = WRITEV (GPMSG 2573)
2574	SYSTEM ROUTINE = OPEN (GPMSG 2574)
2575	SYSTEM ROUTINE = IOCTL (GPMSG 2575)

2576	SYSTEM ROUTINE = WAIT (GPMSG 2576)
2577	SYSTEM ROUTINE = SELECT (GPMSG 2577)
2578	SYSTEM ROUTINE = PAUSE (GPMSG 2578)
2579	SYSTEM ROUTINE = SIGPAUSE (GPMSG 2579)
2580	SYSTEM ROUTINE = SEMOP (GPMSG 2580)
2581	SYSTEM ROUTINE = MSGSND (GPMSG 2581)
2582	SYSTEM ROUTINE = MSGRCV (GPMSG 2582)
2603	WRITING DC RAM WITH ! (GPMSG 2603)
2605	NORMAL SENSE. (GPMSG 2605)
2606	COMPLEMENT SENSE. (GPMSG 2606)

2618	SCAN BOARD DTACK ENABLED. (GPMSG 2618)
2623	LOADING DC RAM WITH ACE COMMANDS. (GPMSG 2623)
2701	INITIALIZING Z MAP. (GPMSG 2701)
2702	INITIALIZING COLOR MAP. (GPMSG 2702)
2708	INITIALIZING FB, CMAP. (GPMSG 2708)
2802	WRITING WRITABLE CONTROL STORE. (GPMSG 2802)
2803	READING WRITABLE CONTROL STORE. (GPMSG 2803)
2804	WRITING CONTROL STORE WITH !. (GPMSG 2804)
2819	WROTE ADDR REG WITH !. (GPMSG 2819)
2847	PATH: ALU(SHIFT R) -> DRAM -> TEMP -> ALU. (GPERR 2847)

2848	PATH: ALU(SHIFT L) -> TEMP -> DRAM -> ALU. (GPERR 2848)
2857	CRC UCODE RESIDENT. (GPMSG 2857)
2860	TAKING CRC OF FRAME BUFFER BOARD !. (GPMSG 2860)
2864	TAKING CRC OF OVERLAY PLANES. (GPMSG 2864)
2865	OPENING FRAME DUMP FILE. (GPMSG 2865)
2868	DUMPING OVERLAY PLANES TO FRAME DUMP FILE. (GPMSG 2868)
2869	DUMPING FRAME BUFFER BOARD ! TO FRAME DUMP FILE. (GPMSG 2869)
2871	P7 P6 P5 P4 P3 P2 P1 P0
2872	EXPECTATION = !
2873	BOARD ! PLANE7..0 = !

2879 OVERLAY PLANE7..0 = !

2880 !

2881 ALL CRC'S VERIFIED. (GPMSG 2881)

2883 CRC DUMP FILE. (GPMSG 2883)

2887 CHIP = ADD. (GPMSG 2887)

2888 CHIP = MULT. (GPMSG 2888)

2889 CHIP = DIV. (GPMSG 2889)

2890 CHIP = Dynamic Test. (Any Floating Point Chip). (GPMSG 2890)

4050**
* INTERACTIVE COLOR MAP CONTROL ROUTINE *
************************************* (GPMSG 4050)

Error Messages

The following is a list of error messages which may appear when using GP3DDIAG. Information messages are listed in the previous section. Other diagnostic error messages may appear at any time. Error messages without the (GPERR #) or (GPMSG #) trailer are generated by the Online Diagnostic subsystem or the Operating system. The "!" indicates that a parameter of some sort replaces the exclamation point when the message is displayed.

5000 FAILED TO CONVERT A NUMBER TO A STRING (GPERR 5000)
CAUSE
ACTION

5001 FRAME BUFFER BUSY TIMEOUT: 3 SEC'S. (GPERR 5001)
CAUSE
ACTION

5002 WINDOW MOVER BUSY. (GPERR 5002)
CAUSE
ACTION

5003 WINDOW MOVER IDLE. (GPERR 5003)
CAUSE
ACTION

5004 SCAN BOARD BUSY. (GPERR 5004)
CAUSE
ACTION

5005 BSCAN BOARD IDLE. (GPERR 5005)
CAUSE
ACTION

5201 STATUS REG BIT ! FAILS. (GPERR 5201)
CAUSE
ACTION

5203 STATUS REG BIT 4 OVERLAY PRESENT BIT FAILS. (GPERR 5203)
CAUSE
ACTION

5205 LGB_ADDR ! EXPECTED ! ACTUAL !. (GPERR 5205)
CAUSE
ACTION

5401 CAN'T OPEN ACE TEST FILE: !. (GPERR 5401)
CAUSE
ACTION

5402 SCAN BUSY STUCK. (GPERR 5402)
CAUSE
ACTION

5403 DC RAM ERROR: ADDR = ! EXPECTED ! GOT !. (GPERR 5403)
CAUSE
ACTION

5404 UNKNOWN CONTROL VALUE IN ACE FILE: !. (GPERR 5404)
CAUSE
ACTION

5413 ZPASS FAILURE. ZPASS SET BUT SHOULD BE 0. (GPERR 5413)
CAUSE
ACTION

5414	ZPASS FAILURE. ZPASS 0 BUT SHOULD BE SET. (GPERR 5414)
CAUSE	
ACTION	

5431	SCREEN CLEAR: BLOCK MOVE BUSY STUCK. (GPERR 5431)
CAUSE	
ACTION	

5701	ERROR OCCURRED WHILE WAITING FOR INITIAL IBF. (GPERR 5701)
CAUSE	
ACTION	

5702	ERROR OCCURRED WHILE WAITING FOR UCODE PARAMETER !. (GPERR 5702)
CAUSE	
ACTION	

5705	ADDRESS = ! RAM ! ACTUAL = ! EXPECTED = !. (GPERR 5705)
CAUSE	
ACTION	

5708	EXPECTED UCODE TERMINATION VALUE OF ! RECEIVED !. (GPERR 5708)
CAUSE	
ACTION	

5710	BOARD ! ABSENT. (GPERR 5710)
CAUSE	
ACTION	

5718	ADDRESS ! EXPECTED ! READ !. (GPERR 5718)
CAUSE	
ACTION	

5725 OVERLAY PLANES ABSENT !. (GPERR 5725)
CAUSE
ACTION

5903 INPUT ERROR. (GPERR 5903)
CAUSE
ACTION

5918 PRIMARY ID = ! EXPECTED $39. (GPERR 5918)
CAUSE
ACTION

5919 SECONDARY ID = ! EXPECTED $4. (GPERR 5919)
CAUSE
ACTION

5922 CHECK SUM = ! EXPECTED $FFFF. (GPERR 5922)
CAUSE
ACTION

5939 FBVEN REG: READ ! EXPECTED !. (GPERR 5939)
CAUSE
ACTION

5946 FBWEN, OPWEN ! WROTE ! READ ! EXPECTED !. (GPERR 5946)
CAUSE
ACTION

5947 DRIVE ! READ ! EXPECTED !. (GPERR 5947)
CAUSE
ACTION

6201	NON SYSTEM CALL IN TEST INTERRUPTED. (GPERR 6201)
CAUSE	
ACTION	

6202	SYSTEM CALL IN TEST INTERRUPTED. (GPERR 6202)
CAUSE	
ACTION	

6203	UCODE REQUIRES > 16 BANKS OF CONTROL STORE. (GPERR 6203)
CAUSE	
ACTION	

6204	SIGNAL 'SIGPWR' TRAPPED. NO ACTION TAKEN. (GPERR 6204)
CAUSE	
ACTION	

6205	SIGNAL NUMBER ! TRAPPED. (GPERR 6205)
CAUSE	
ACTION	

6206	SIGNAL ! TRAPPED. PROGRAM ABORTED. (GPERR 6206)
CAUSE	
ACTION	

6207	WARNING ... SIGNAL ENABLE TRIED BEFORE DISABLE. (GPERR 6207)
CAUSE	
ACTION	

6208	ERROR IN ARM SIGNAL TRAPPING. (GPERR 6208)
CAUSE	
ACTION	

6209 FRAME BUFFER DEVICE FILE NEVER OPENED. (GPERR 6209)
CAUSE
ACTION

6210 BUS ERROR ENCOUNTERED WHILE WRITING FRAME BUFFER ENABLE REGISTER ON INTERFACE CARD. UNABLE TO UNLOCK FRAME BUFFER SPACE. PLEASE POWER DOWN THIS RENAISSANCE AND ENTER A CARRIAGE RETURN. FAILURE TO DO SO WILL CAUSE OTHER TESTS RUNNING ON THIS SPU TO FAIL. (GPERR 6210)
CAUSE
ACTION

6211 NON-NUMERIC COMMAND LINE ARGUMENT. (GPERR 6211)
CAUSE
ACTION

6212 CONTROL SPACE NUMBER FIELD. (GPERR 6212)
CAUSE
ACTION

6213 COMMAND LINE ARGUMENT OUT OF RANGE. (GPERR 6213)
CAUSE
ACTION

6214 CONTROL SPACE NUMBER 1..7. (GPERR 6214)
CAUSE
ACTION

6215 CAN'T OPEN CONTROL SPACE DEVICE FILE !. (GPERR 6215)
CAUSE
ACTION

6217 CAN'T GCMAP CONTROL/FRAMEBUFFER SPACE ERRNUM = !. (GPERR 6217)
CAUSE
ACTION

6220 FILE READ ERROR. READ ! BYTES, EXPECTED !, UADDR !. (GPERR 6220)
CAUSE
ACTION

6221 ERROR OPENING UCODE FILE !. (GPERR 6221)
CAUSE
ACTION

6227 SIGNAL ENABLE TRIED BEFORE DISABLE. (GPERR 6227)
CAUSE
ACTION

6237 WARNING .. OVERLAY RELOCATED TO FRAME BUFFER 0 LOCATION. (GPERR 6237)
CAUSE
ACTION

6238 CAN'T OPEN CRC REFERENCE FILE !. (GPERR 6238)
CAUSE
ACTION

6239 ERROR ENCOUNTERED READING CRC FILE ! NUMBER READ !. (GPERR 6239)
CAUSE
ACTION

6240 ERROR OCCURRED AT CRC NUMBER ! AND BOARD NUMBER !. (GPERR 6240)
CAUSE
ACTION

6241 ONLY ! CRC'S IN FILE ! EXPECTED !. (GPERR 6241)
CAUSE
ACTION

6242 TEMPERATURE MEASURING DISABLED: FILE 'TEMPERATURE' MISSING. (GPERR 6242)
CAUSE
ACTION

6243 TEMPERATURE DATA IN FILE 'TEMPERATURE' IS OUT OF DATE. (GPERR 6243)
CAUSE
ACTION

6255 LONG JUMP ERROR IN SIGNAL HANDLER. (GPERR 6255)
CAUSE
ACTION

6257 ILLEGAL RPT VALUE IN STRPRT. (GPERR 6257)
CAUSE
ACTION

6258 STRING OVERFLOW IN STRRPT. (GPERR 6258)
CAUSE
ACTION

6259 NULL FOPEN VALUE IN OPEN_PRINTER. (GPERR 6259)
CAUSE
ACTION

6260 CALLED CLOSED PRINTER. PRINTER NOT OPEN. (GPERR 6260)
CAUSE
ACTION

6261 SYSTEM ERROR IN CLOSE_PRINTER. (GPERR 6261)
CAUSE
ACTION

6262 UNSUCCESSFUL DOWN OF FRAME BUFFER SEMAPHORE. (GPERR 6262)
CAUSE
ACTION

6263 UNSUCCESSFUL UP OF FRAME BUFFER SEMAPHORE. (GPERR 6263)
CAUSE
ACTION

6264 DEVICE FILE IS LOCKED. (GPERR 6264)
CAUSE
ACTION

6265 INVALID CONFIGURATION COMMAND LINE ARGUMENT. (GPERR 6265)
CAUSE
ACTION

6283 SYSTEM ROUTINE = UNKNOWN (GPERR 6283)
CAUSE
ACTION

6519 ACE STATUS REG INDICATES ACE BUSY. (DID NOT HALT). (GPERR 6519)
CAUSE
ACTION

6520 ACE NOT BUSY AT COUNT !. (GPERR 6520)
CAUSE
ACTION

6521 ACE STILL BUSY AFTER ! COUNTS. (GPERR 6521)
CAUSE
ACTION

6522 ACE REGISTER ! CONTAINS ! EXPECTED !. (GPERR 6522)
CAUSE
ACTION

6526 ADDRESS = ! DATA = ! !. (GPERR 6526)
CAUSE
ACTION

6527 ADDRESS = ! ACTUAL = ! EXPECTED = !. (GPERR 6527)
CAUSE
ACTION

6701 SEMOP ERRNO = EINVAL. (GPERR 6701)
CAUSE
ACTION

6702 SEMOP ERRNO = EFBIG. (GPERR 6702)
CAUSE
ACTION

6703 SEMOP ERRNO = E2BIG. (GPERR 6703)
CAUSE
ACTION

6704 SEMOP ERRNO = EACCES. (GPERR 6704)
CAUSE
ACTION

6705 SEMOP ERRNO = EAGAIN. (GPERR 6705)
CAUSE
ACTION

6706	SEMOP ERRNO = ENOSPC. (GPERR 6706)
CAUSE	
ACTION	

6707	SEMOP ERRNO = ERANGE. (GPERR 6707)
CAUSE	
ACTION	

6708	SEMOP ERRNO = EFAULT. (GPERR 6708)
CAUSE	
ACTION	

6709	FRAME BUFFER SEMAPHORE EXISTS ALREADY. (GPERR 6709)
CAUSE	
ACTION	

6710	COULD NOT SEMGET SEMAPHORE. (GPERR 6710)
CAUSE	
ACTION	

6711	ERROR WHILE INITIALIZING FRAME BUFFER SEMAPHORE (DOWN). (GPERR 6711)
CAUSE	
ACTION	

6712	ERROR WHILE INITIALIZING FRAME BUFFER SEMAPHORE REF COUNT. (GPERR 6712)
CAUSE	
ACTION	

6714	! UNSUCCESSFUL ATTEMPTS TO ACQUIRE FRAME BUFFER SEMAPHORE. (GPERR 6714)
CAUSE	
ACTION	

6715 ERROR UPING FRAME SEMAPHORE IN DESTROY_FRAME_SEMAPHORE. (GPERR 6715)
CAUSE
ACTION

6716 ERROR DOWNING FRAME SEMAPHORE IN DESTROY_FRAME_SEMAPHORE. (GPERR 6716)
CAUSE
ACTION

6717 ! TEST PROCESSES STILL RUNNING. (GPERR 6717)
CAUSE
ACTION

6718 DESTROYED FRAME BUFFER SEMAPHORE. (GPERR 6718)
CAUSE
ACTION

6719 UNKNOWN ERRNO IN DOWN_FRAME_SEMAPHORE. (GPERR 6719)
CAUSE
ACTION

6720 ERROR WHILE INITIALIZING FRAME BUFFER SEMAPHORE (UP). (GPERR 6720)
CAUSE
ACTION

6721 ERROR REFERENCING SEMAPHORE CRC REFERENCE FILE. (GPERR 6721)
CAUSE
ACTION

6722 ERROR UPING FRAME BUFFER SEMAPHORE. (GPERR 6722)
CAUSE
ACTION

6750 UNKNOWN BOOLEAN VALUE IN MAIN.C. (GPERR 6750)
CAUSE
ACTION

6751 NUMBER OF TESTS OVERFLOW IN TAF.C. (GPERR 6751)
CAUSE
ACTION

6752 NULL FOPEN VALUE IN OPEN_DATA_BASE(). (GPERR 6752)
CAUSE
ACTION

6803 CD FULL FLAG NEVER CLEARED. (GPERR 6803)
CAUSE
ACTION

6804 DAT OR SPININIT FILE FAILED TO OPEN. (GPERR 6804)
CAUSE
ACTION

6805 DAT OR SPININIT FILE FAILED TO CLOSE. (GPERR 6805)
CAUSE
ACTION

6806 SPINONCE FILE FAILED TO OPEN. (GPERR 6806)
CAUSE
ACTION

6807 SPINONCE FILE FAILED TO CLOSE. (GPERR 6807)
CAUSE
ACTION

6809 SPIN CRC'S NOT DONE, CD FULL FLAG TIMEOUTS OCCURRED. (GPERR 6809)
CAUSE
ACTION

7101 UCODE DID NOT COMPLETE TIME MEASUREMENT RESULTS INVALID. (GPERR 7101)
CAUSE
ACTION

7102 COLOR MAP BLINK BIT (BIT5) LL = ! ACTUAL = ! UL = !. (GPERR 7102)
CAUSE
ACTION

7103 COLOR MAP NOT COPY BIT (BIT2) LL = ! ACTUAL = ! UL = !. (GPERR 7103)
CAUSE
ACTION

7104 COLOR MAP H BLANK BIT (BIT1) LL = ! ACTUAL = ! UL = !. (GPERR 7104)
CAUSE
ACTION

7105 COLOR MAP V SYNC BIT (BIT0) LL = ! ACTUAL = ! UL = !. (GPERR 7105)
CAUSE
ACTION

7106 VERTICAL BLANK IN PROGRESS BIT(BIT0) LL = ! ACTUAL = ! UL = !. (GPERR 7106)
CAUSE
ACTION

7107 INTV DID NOT RESET. (GPERR 7107)
CAUSE
ACTION

7108 INTV DID NOT SET. (GPERR 7108)
CAUSE
ACTION

7110 WBUSY FAST MOVE FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7110)
CAUSE
ACTION

7111 INTB FAST MOVE FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7111)
CAUSE
ACTION

7112 WBUSY SLOW MOVE FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7112)
CAUSE
ACTION

7113 INTB SLOW MOVE FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7113)
CAUSE
ACTION

7114 WBUSY SLOW MOVE DEST FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7114)
CAUSE
ACTION

7115 INTB SLOW MOVE DEST FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7115)
CAUSE
ACTION

7116 WBUSY SLOW MOVE PATT FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7116)
CAUSE
ACTION

7117 INTB SLOW MOVE PATT FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7117)
CAUSE
ACTION

7118 ACEBUSY TEST FAILED: LL = ! ACTUAL = ! UL = !. (GPERR 7118)
CAUSE
ACTION

7301 BANK ! UADDR ! LONGWORD ! EXPECTED ! READ !. (GPERR 7301)
CAUSE
ACTION

7305 IBF NOT SET. (TIMEOUT AFTER ! SEC'S. (GPERR 7305)
CAUSE
ACTION

7306 FRAME DUMP (!,!) HEIGHT = ! DRIVE = ! MASK = ! (GPERR 7306)
CAUSE
ACTION

7309 TRANSFORM ENGINE WON'T HALT. (GPERR 7309)
CAUSE
ACTION

7310 TRANSFORM ENGINE WON'T CONTINUE. (GPERR 7310)
CAUSE
ACTION

7311 HALT NOT SET. TIMED OUT AFTER ! SEC'S. (GPERR 7311)
CAUSE
ACTION

7313 ERROR NUMBER !. (GPERR 7313)
CAUSE
ACTION

7314 ERROR NUMBER OUT OF RANGE. (GPERR 7314)
CAUSE
ACTION

7315 READ ! EXPECTED !. (GPERR 7315)
CAUSE
ACTION

7318 COULD NOT READ ERROR PARAMETER !. (GPERR 7318)
CAUSE
ACTION

7322 ADDRESS !. (GPERR 7322)
CAUSE
ACTION

7324 READ !. (GPERR 7324)
CAUSE
ACTION

7326 EXPECTED !. (GPERR 7326)
CAUSE
ACTION

7328 `READ ADDRESS REG AND FOUND   !. (GPERR 7328)`
CAUSE
ACTION

7329 `DATA READ  !. (GPERR 7329)`
CAUSE
ACTION

7331 `DATA RAM ERROR: UPPER 16 BITS. (GPERR 7331)`
CAUSE
ACTION

7332 `DATA RAM ERROR: LOWER 16 BITS. (GPERR 7332)`
CAUSE
ACTION

7333 `DATA RAM ADDRESS REG ERROR. (GPERR 7333)`
CAUSE
ACTION

7334 `DATA RAM ADDRESSING ERROR DURING READ. (GPERR 7334)`
CAUSE
ACTION

7335 `DATA RAM ADDRESSING ERROR DURING WRITE. (GPERR 7335)`
CAUSE
ACTION

7336 `UNEXPECTED ERROR NUMBER. (GPERR 7336)`
CAUSE
ACTION

7337 `PRAM ERROR WITH UCODE ADDRESSING. (GPERR 7337)`
CAUSE
ACTION

7338 PRAM ERROR WITH PRAR ADDRESSING. (GPERR 7338)
CAUSE
ACTION

7340 ILLEGAL PARAMETER 1 VALUE. (GPERR 7340)
CAUSE
ACTION

7341 CALCULATION NUMBER !. (GPERR 7341)
CAUSE
ACTION

7342 ADDRESS OF INCORRECT RESULT !. (GPERR 7342)
CAUSE
ACTION

7349 INVALID PATH READ FOR PARAMETER 1. (GPERR 7349)
CAUSE
ACTION

7350 VALUE WRITTEN = !. (GPERR 7350)
CAUSE
ACTION

7351 VALUE READ = !. (GPERR 7351)
CAUSE
ACTION

7352 EXPECTED FLOATING POINT EXCEPTION. (GPERR 7352)
CAUSE
ACTION

7353 UNEXPECTED FLOATING POINT EXCEPTION. (GPERR 7353)
CAUSE
ACTION

7354 INCORRECT FLOATING POINT RESULT: INDIVIDUAL CHIP TEST. (GPERR 7354)
CAUSE
ACTION

7355 INCORRECT FLOATING POINT RESULT: MULTI-CHIP TEST. (GPERR 7355)
CAUSE
ACTION

7356 ERROR READING FLOATING POINT TEMP REG. (GPERR 7356)
CAUSE
ACTION

7358 SIGNATURE [!] = !. (GPERR 7358)
CAUSE
ACTION

7366 FILE EXISTS ALREADY. NO FRAME DUMP DONE. (GPERR 7366)
CAUSE
ACTION

7367 UNEXPECTED OPEN ERRNO OF ! RECEIVED. (GPERR 7367)
CAUSE
ACTION

7370 INCORRECT CRC DETECTED. (GPERR 7370)
CAUSE
ACTION

7382	CAN'T OPEN CRC DUMP FILE !. (GPERR 7382)
CAUSE	
ACTION	
7391	ERROR IN REGISTER OPERATIONS. (GPERR 7391)
CAUSE	
ACTION	
7392	ERROR IN REGISTER WRITE OPERATIONS. (GPERR 7392)
CAUSE	
ACTION	
7393	ERROR IN Q REGISTER SHIFT RIGHT. (GPERR 7393)
CAUSE	
ACTION	
7394	ERROR IN Q REGISTER SHIFT LEFT. (GPERR 7394)
CAUSE	
ACTION	
7395	REGISTER OPERATION FAILURE. (GPERR 7395)
CAUSE	
ACTION	
7396	$8888+$7777+CARRY PRODUCED BIT 15 SET FAILURE. (DELAYED). (GPERR 7396)
CAUSE	
ACTION	
7397	$8888+$7777+CARRY PRODUCED BIT 15 FAILURE. (NOT DELAYED). (GPERR 7397)
CAUSE	
ACTION	

7398 $7777+$7777+CARRY DIDN'T PRODUCE OVERFLOW (DELAYED). (GPERR 7398)
CAUSE
ACTION

7399 $8888+$8888+CARRY DIDN'T PRODUCE OVERFLOW (DELAYED). (GPERR 7399)
CAUSE
ACTION

7401 $0000+$7777+CARRY PRODUCED OVERFLOW (DELAYED). (GPERR 7401)
CAUSE
ACTION

7402 (0+R0) $0000+$8888+CARRY DID NOT PRODUCE F15. (GPERR 7402)
CAUSE
ACTION

7403 $0000+$8888+CARRY DID NOT PRODUCE F15 (DELAYED). (GPERR 7403)
CAUSE
ACTION

7404 $8888+$8888+CARRY DID NOT PRODUCE COUT (DELAYED). (GPERR 7404)
CAUSE
ACTION

7405 $7777+$7777+CARRY DID PRODUCE COUT (DELAYED). (GPERR 7405)
CAUSE
ACTION

7406 $8888+$7777+NOCARRY DID PRODUCE COUT (DELAYED). (GPERR 7406)
CAUSE
ACTION

7407 CJP PASS INSTRUCTION DID NOT BRANCH. (GPERR 7407)
CAUSE
ACTION

7408 CJP FAIL INST BRANCHED WHEN IT SHOULDN'T HAVE. (GPERR 7408)
CAUSE
ACTION

7409 JRP PASS INSTRUCTION DID NOT BRANCH. (GPERR 7409)
CAUSE
ACTION

7410 JRP PASS INST BRANCHED TO (REGISTER) NOT TO (PIPE). (GPERR 7410)
CAUSE
ACTION

7411 JRP FAIL INSTRUCTION DID NOT BRANCH TO (REGISTER). (GPERR 7411)
CAUSE
ACTION

7412 JRP FAIL INSTRUCTION BRANCHED TO (PIPE) NOT (REG.). (GPERR 7412)
CAUSE
ACTION

7413 CJS PASS DID NOT CHANGE PROGRAM FLOW. IT SHOULD HAVE. (GPERR 7413)
CAUSE
ACTION

7414 CJPP PASS INSTRUCTION DID NOT BRANCH AS REQUIRED. (GPERR 7414)
CAUSE
ACTION

7415 CJS FAIL INSTRUCTION DID BRANCH; IT SHOULD NOT HAVE. (GPERR 7415)
CAUSE
ACTION

7416 CRTN FAIL INSTRUCTION SHOULD NOT HAVE RETURNED. (GPERR 7416)
CAUSE
ACTION

7417 CRTN PASS DID NOT RETURN. (GPERR 7417)
CAUSE
ACTION

7418 CJPP FAIL INSTRUCTION BRANCHED BUT SHOULDN'T HAVE DONE SO. (GPERR 7418)
CAUSE
ACTION

7419 JSRP PASS INSTRUCTION BRANCHED TO REGISTER/SHOULD BE PIPE. (GPERR 7419)
CAUSE
ACTION

7420 JSRP FAIL INSTRUCTION BRANCHED TO PIPE/SHOULD BE REGISTER. (GPERR 7420)
CAUSE
ACTION

7421 RPCT COUNTER=0..SHOULD NOT HAVE BRANCHED. (GPERR 7421)
CAUSE
ACTION

7422 TWB FAIL SHOULD HAVE BRANCHED BUT DID NOT DO SO. (GPERR 7422)
CAUSE
ACTION

7423 TWB PASS SHOULD NOT HAVE BRANCHED BUT DID. (GPERR 7423)
CAUSE
ACTION

7424 JP INSTRUCTION SHOULD HAVE JUMPED BUT DID NOT. (GPERR 7424)
CAUSE
ACTION

7426 ADD WITH CARRY TO RO FAILED TO LOAD RO WITH ZERO. (GPERR 7426)
CAUSE
ACTION

7427 TE, ALU, LOAD RO WITH 0000H FAILED. (GPERR 7427)
CAUSE
ACTION

7428 RFCT INSTRUCTION FAILED WITH 11 LOADED INTO COUNTER. (GPERR 7428)
CAUSE
ACTION

7429 RPCT INSTRUCTION FAILED WITH 0 LOADED INTO COUNTER. (GPERR 7429)
CAUSE
ACTION

7430 TWB INSTRUCTION FAILED WITH 11 LOADED IN COUNTER CC FALSE. (GPERR 7430)
CAUSE
ACTION

7431 RFCT INSTRUCTION FAILED WITH 11 LOADED INTO COUNTER FROM IDB. (GPERR 7431)
CAUSE
ACTION

7432 LOOP INSTR. 11 LOADED INTO COUNTER & FROM IDB. (FEZD COND FIRST USE). (GPERR 7432)

CAUSE

ACTION

7433 2910 GENERATED SFULL TOO SOON. CHECK AM2910. SHOULD HAVE 'A' SUFFIX. (GPERR 7433)

CAUSE

ACTION

7434 2910 DID NOT GENERATE SFULL WHEN IT SHOULD HAVE. (GPERR 7434)

CAUSE

ACTION

7435 RPCT COUNTER<>0 SHOULD HAVE BRANCHED BUT DIDN'T. (GPERR 7435)

CAUSE

ACTION

7436 UNKNOWN BD # IN DUMP_FRAME_BOARD. (GPERR 7436)

CAUSE

ACTION

7437 FILE WRITE ERROR IN DUMP_FRAME_BUFFER_BOARD. (GPERR 7437)

CAUSE

ACTION

7500

```
**********************
* BUS ERROR TRAPPED  *
***************** (GPERR 7500)
```

CAUSE

ACTION

8

HP 9000 Series 300 Graphics Diagnostics

For information on the **g98705dg** diagnostic for the HP 98705A/B/C Graphics Processor, see chapter 8 of the *HP 98705A/B/C Graphics Processor Hardware Support Manual (part no. 98705-90030).*

For information on the **g98735dg** diagnostic for the HP 98735/6A/6B Graphics Display Controller, see chapter 8 of the *HP 98735/6A/6B Graphics Display Controller Hardware Support Manual (part no. 98735-90030).*

READER COMMENT SHEET

Systems Technology Division

Precision Architecture RISC:
Support Tools Manual (Licensed Users)
Vol. 1, SPU

Manual Part Number 5960-3149 December 1992

A reader comment sheet helps us to improve the readability and accuracy of the document. It is also a vehicle for recommending enhancements to the product or manual. Please use it to suggest improvements.

SERIOUS ERRORS, such as technical inaccuracies that may render a program or a hardware device inoperative, should be reported to your HP Response Center or directly to a Support Engineer. An engineer will enter the problem on HP's STARS (Software Tracking and Reporting System). This will ensure that critical and serious problems receive appropriate attention as soon as possible.

Editorial suggestions (please include page numbers): ______________________

____

____

____

Recommended improvements (attach additional information, if needed): ____________

____

____

____

Name: ______________________ **Date:** __________

Job Title: ______________________ **Phone:** __________

Company: ______________________________

Address: ______________________________

____

☐ Check here if you would like a reply.

Hewlett-Packard has the right to use submitted suggestions without obligation, with all such ideas becoming property of Hewlett–Packard.

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070, CUPERTINO, CA

– POSTAGE WILL BE PAID BY ADDRESSEE –

Publications Manager – Hardware
Hewlett-Packard Company
Systems Technology Division
19483 Pruneridge Avenue
Mailstop 42UK
Cupertino, California 95014-9786

Fold and Tape

Support Tools Manual (Licensed Users)
Volume 2, Device Adapters/Muxes

PA-RISC Computer Systems

HEWLETT
PACKARD

HP Part No. 5960-3151
Printed in USA December 1992

Edition 1 - E1292

Notice

System Technology Division
19483 Pruneridge Avenue
Cupertino, CA 95014

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

First Edition . . . . December 1992

List of Effective Pages

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. No information is incorporated into a reprinting unless it appears as a prior update.

Safety and Regulatory Information

Safety Considerations

This documentation must be reviewed for familiarization with safety markings and instructions before operation. The following figures show some of the safety symbols used in the documentation to indicate various safety considerations.

Warning

The WARNING sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not done correctly or adhered to, could result in injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.

Caution

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not done correctly or adhered to, could damage or destroy part or all of the product. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

Preface

This manual contains information about the device adapter and multiplexer diagnostics currently available for PA-RISC computer systems. It is intended to be used as technical support hardware documentation by licensed users only. The procedures and software described are focused primarily on the hardware troubleshooting environment and require specific training for correct and safe usage.

Contents

1

AFI Device Adapter Diagnostic

Introduction

The AFI Device Adapter Diagnostic (Asynchronous FIFO Interface Device Adapter Diagnostic, `AFIDAD`) will test the HP 27114A AFI. This diagnostic runs on any HP 9000 Series 800 computer system. `AFIDAD` is part of the Online Diagnostic Subsystem and will:

- Identify the product type and the hardware revision code number.
- Report the status of the AFI card.
- Reset the AFI card.
- Test the majority of the circuits on the AFI card.
- Perform a loopback test of AFI circuits, including the frontplane interface circuitry. This test requires the use of the loopback test hood.
- Allow the user to directly control the state machine on the AFI card.

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10014.

Minimum Configuration

The hardware required to run `AFIDAD` consists of an HP 9000 Series 800 computer, an AFI card, and a loopback test hood. Required software includes an HP-UX operating system, the AFI driver, and the Online Diagnostic subsystem, of which `AFIDAD` is a part. All of this software is contained on the Fundamental Operating System (FOS) tape, and is automatically installed when the tape is first read on your computer system. Contact your system manager if the appropriate software is not present on your system.

Operating Instructions

Sections 2, 4. 5, and 7 of `AFIDAD` are destructive and require security 1. Refer to the section on the DUI for information on the available security levels and test modes, and how each are determined.

Default Tests

If you do not specify sections to be run, the single default section, Section 3, will be executed.

Run Command

To bring up the On-Line Diagnostic subsystem, enter the following command to the HP-UX system prompt:

```
% sysdiag
```

The Diagnostic subsystem responds with the following prompt indicating that diagnostic system access has been granted to the user:

```
DUI >
```

Typing `HELP` causes a summary of the DUI and its commands to be printed. Refer to the DUI Section of this manual for details.

Note The device to be tested must be powered up and on line. Device physical locations (pdev) shown in the RUN commands are those of the devices on the "typical A1002A" system configuration described in the chapter on DUI. The pdev value entered must be correct for the system being tested.

For example, to run the diagnostic, you might enter:

```
DUI >     run afidad pdev=4.5 <RUN Command Options>
                          |          |
                          |     none required for
                          |     default test suite
                          |
                 insert physical location of
                 device adapter to be tested here,
                 or type the devfile name
```

The user must specify either a physical device number PDEV or a logical device number LDEV as part of the run command string. The LDEV or the diagnostic device file for the respective AFI card must be present if the AFI card is specified via a diagnostic device file name. Diagnostic device file name for the AFI is usually called gpio0, gpiol, ...etc., and can be found in /dev/diag as are other diagnostic device file names on the HP-UX OS.

Test Execution

If the system is unable to grant access to the AFI, the Online Diagnostic subsystem prints an error message. (The **AFIDAD** diagnostic will not output an error message, and will terminate.) When the system is able to grant access to the AFI, a banner and welcome message will be displayed.

Any possible errors are reported when they are detected. The section stops as soon as an error is detected. Most errors in **AFIDAD** are descriptive such that they can be used to trace to a specific failure. If a section executes without error, it is so reported.

If you enter **HELP**, a general description of the **AFIDAD** diagnostic will be displayed. If you enter the **More Help** section, **AFIDAD** will display a description of the specified section.

Test Descriptions

There are seven diagnostic sections available in **AFIDAD**. There are no steps in any section of **AFIDAD**. There is only one default section: Section 3, Identify. You may select any of the other sections to execute when you run **AFIDAD**.

Section 1	More Help
Section 2	Reset
Section 3	Identify
Section 4	Hardware Test
Section 5	Loopback Test
Section 6	Status
Section 7	Register Level Input/Output Transactions

Section 1—MORE HELP

The More Help Section prompts you for the section which needs more description. A (Return) terminates this section. If you specify a wrong section number, it will be rejected and you will be prompted for another section number.

Section 1 is normal and is not in the default set.

Section 1 will output the following message:

```
Section 1 -- More Help

   This Section allows you to get more information on any of the
   sections [1..7] of this diagnostic.  Please indicate the number
   of the section for which you require more information.
   Entering a lone <CR> to the prompt exits this section.

   More Help (1..7, <CR>) :

End of Section 1 -- More Help
```

Section 2—RESET

The Reset Section is used to perform a complete reset of the AFI card. It has the same effect on the AFI card as a power on of the host computer (i.e., the card is reset and the self-test is downloaded from the host computer and executed).

Section 2 is destructive and is not in the default set.

If no errors are generated, section 2 will output the following message:

```
Section 2 -- Reset

   NO ERRORS DETECTED while resetting the device adapter.

End of Section 2 -- Reset
```

Section 3—IDENTIFY

The Identify Section is the default section executed whenever **AFIDAD** is run and is used to identify the card under test. The test is aborted if the identified card is not an AFI card. Other information reported includes the device adapter manager revision number and hardware code revision code number.

Section 3 is normal and is the only section in the default set.

If no errors are generated, section 3 will output the following message:

```
Section 3 -- Identify

   CIO card ID byte  = 32
   Hardware Revcode  = ?
   DAM available for hardware revision ?

End of Section 3 -- Identify
```

Section 4—HARDWARE TEST

The Hardware Test Section exercises the majority of circuits on the AFI card. The only circuits not covered are the frontplane interface circuits (these circuits are exercised by Section 5).

Section 4 is destructive and is not in the default set.

If no errors are generated, section 4 will output the following message:

```
Section 4 -- Hardware Test

   CIO card ID byte         = 32
   Hardware Revcode         = ?
   DAM available for hardware revision ?

   No hardware errors found.

End of Section 4 -- Hardware Test
```

Section 5—LOOPBACK TEST

The Loopback Test Section exercises all testable circuits on the AFI card, including the fronplane interface circuits. It checks all line drivers and receivers. This section requires the use of the AFI card loopback test hood.

Section 5 is destructive and is not in the default set.

If no errors are generated, section 5 will output the following message:

```
Section 5 -- Hardware test with test hood

   CIO card ID byte      = 32
   Hardware Revcode      = ?
   DAM available for hardware revision ?
```

```
   No hardware errors found.

End of Section 5 -- Hardware test
```

Section 6—STATUS

The Status section reports the current conditions of the AFI card. This information includes the:

- condition of the ARQ and ARQ Enable flip-flops
- condition of parity enable
- conditions of FIFO
- states of frontplane handshake signals PFLAG and PCTL
- presence or absence of test hood
- states of the peripheral status lines

Section 6 is normal and is not in the default set.

If no errors are generated, Section 6 will output the following message:

```
Section 6 -- Status

  ARQ interrupt is pending/clear.

  ARQ interrupt is enabled/disabled.

  CEND is asserted/de-asserted.

  FIFO has room for data/is full.

  FIFO has data in it/no data in it.

  PFLG is asserted/de-asserted.

  PCTL is asserted/de-asserted.

  Transfer counter equals zero/is not equal to zero.

  Transfer counter value is ?

  ATTN is reset/set.

  DEND is reset/set.

  FIFO contains approximately
 0 - 8  words
 9 - 31 words
32 - 55 words
56 - 64 words

  State of status lines STS0 through STS5 (in that order):
      Asserted
      De-asserted

End of Section 6 -- Status
```

Section 7—REGISTER LEVEL INPUT/OUTPUT TRANSACTIONS

The Control Section allows you to directly control the state machine on the AFI card. You are prompted for a new control value to be output to the card. Illegal values (such as out of range, etc.) are rejected and you are prompted for another value. Once a value is accepted and output to the card, the current status of the AFI card is also reported in an abbreviated form. You should have a good working knowledge of the AFI card to get any worthwhile results from using this section.

Section 7 is destructive and is not in the default set.

If no errors are generated, section 7 will output the following message:

```
Section 7 -- Register Level Input/Output Transactions

   Enter register selection , read/write and output data (if applicable)
   User register 2 for poll function.  Last entered value for same
   question is repeated if carriage return is entered.

   Register number (0, 1, 2 [POLL], 3, 7, 9, 10, 11, exit) =

   Input 0 for read, 1 for write:

End of Section 7 -- Register Level Input/Output Transactions
```

Error Messages

The following is a list of error messages which may appear when using **AFIDAD**. Other error messages may occur which do not have the **AFIDAD** # trailer; these messages are generated by the Online Diagnostic Subsystem or the operating system. Consult the DUI section of this manual and the operating system manuals for these errors. The "!" indicates that a parameter of some sort will replace the exclamation point when the message is displayed.

99 THE MAXIMUM NUMBER OF ERROR MESSAGES HAS BEEN EXCEEDED. (AFIDADERR 99)

CAUSE

ACTION

---

202 AN ERROR OCCURRED DURING RESET. (AFIDADERR 202) AFI_DAR STATUS RETURNED = !

CAUSE

ACTION

1XXYY THE FOLLOWING ERROR IS NUMBERED AS 1XXYY WHERE XX IS STEP AND YY IS SUB-STEP HARDWARE ERROR DETECTED (AFIDADERR !)

CAUSE

ACTION

-1 RESOURCE ALLOCATION ERROR IN SYSTEM. (AFIDADERR -1) TEST IS TERMINATED

CAUSE

ACTION

2

PSI Device Adapter Diagnostic

Introduction

PSIDAD tests Programmable Serial Interface cards on an HP Precision Architecture RISC computer system which supports the Online Diagnostic subsystem.

Defects and Enhancements

Submit defect reports and enhancement requests for this diagnostic through the STARS database referencing Product Number 30600-10027.

Minimum Configuration

The minimum configuration required to run this diagnostic consists of an HP Precision Architecture RISC computer system up and running on either the MPE/iX or HP-UX operating system.

Operating Instructions

The PSIDAD Diagnostic is accessed by the user via the Diagnostic User Interface.

Default Tests

If you do not specify sections and steps to be run, the following default sections and steps will be executed depending on the current mode of the system:

Section 3	Identify
Section 5	Selftest
Section 6	Status

RUN Command

To bring up the Online Diagnostic subsystem, enter the following command to the system prompt:

```
sysdiag
```

The diagnostic subsystem responds with the following prompt indicating that access has been granted to the user:

```
DUI >
```

Typing HELP causes a summary of the DUI function and its commands to appear on the screen.

Note The device to be tested must be powered up and on line. The physical device location (pdev) shown below is only an example. The pdev value entered must be correct for the system being tested.

For example, to run the diagnostic in an MPE/iX environment, you might enter:

```
DUI > RUN PSIDAD pdev=24 <RUN Command Options>
                       |          |
                       |   none required for
                       |   default test suite
                       |
              insert physical location of
              device adapter to be tested here;
              alternatively, for MPE/iX,
              type the ldev number;
              for HP-UX, type the devfile name
              (e.g., dev = /dev/diag/psi0)
```

All parameters available in the RUN command are acceptable as parameters when running this diagnostic.

Note For the 2.0 release, the manner in which steps are designated for execution with the run command has changed; for example, from `section = 9 step = 92` to `section = 9(92)`.

Test Execution

When `PSIDAD` is run, a header and welcome message will be displayed.

Test Section Descriptions

The following test sections are available with `PSIDAD`:

Section 1	More Help
Section 2	Reset
Section 3	Identify
Section 5	Selftest
Section 6	Status
Section 8	Internal Hardware
Section 9	External Hardware
Section 10	Manufacturing Utilities
Section 15	EEPROM Failure History (HP-PB only)

Note

For the 2.0 release version of this diagnostic, there is no **More Help** section. Help is obtained by entering **help psidad** at the `DUI` prompt.

Section 1 - More Help

This section allows the user to obtain more information about a particular section or step in PSIDAD. The security level is normal. This is an interactive section which asks for the number of the section for which more information is desired. To exit this section, simply enter (Return) to the More Help prompt.

Note For the 2.0 release version of this diagnostic, there is no **More Help** section. Help is obtained by entering **help psidad** at the DUI prompt.

Output:

```
    Section 1 -- More Help

This section allows you to get more information on all of
the sections [1..10] of this diagnostic.  Please indicate the
number of the section for which you need more information.
Entering a <return> to the prompt exits this section.

More Help >>

    End of Section 1 -- More Help
```

Section 2 - Reset

Section 2 brings the PSI Device Adapter into an operational (power-on) state, clearing any residual error conditions. The security level is destructive.

Output:

```
Section 2 -- Reset

PSI card and driver successfully reset.

End of Section 2 -- Reset
```

Section 3 - Identify

Section 3 conveys static information about the hardware, firmware, driver, and DAR. The security level is normal.

Output:

```
Section 3 -- Identify

Hardware version:
  Hardware model = $4
  Hardware revision  = $0
Software version:
  Software model = $20
  Software revision = $0
  Software option = $20
IO_DC revision = $0
First EPROM part number = 1818-xxxx; date code = 2808
Second EPROM part number = 1818-xxxx; date code = 2808
RAM starting address = $FFFC0000; size = $4000
DAR version = A.00.00; driver ID = 77

End of Section 3 -- Identify
```

Section 5 - Selftest

Section 5 verifies that the board hardware is generally operational. The security level is destructive.

Output:

For Midbus, the following output is typical:

```
Section 5 -- Selftest

Selftest passed.

End of Section 5 -- Selftest
```

For HP-PB, the following output is typical:

```
Section 5 -- Selftest

Performing a reset_HT...

Performing a reset...

Performing reads and writes to PSI card...

Performing a selftest...

Selftest passed.

End of Section 5 -- Selftest
```

Section 6 - Status

Section 6 conveys information about the current dynamic state of the board. The security level is normal.

Output:

The following numbers are not real and are used only as examples.

```
Section 6 -- Status

ROM firmware currently running.
Current firmware part number = 1818-xxxx; date code = 2808
Current firmware version: model = $6; revision = $0; option = $2
Cable type connected: No cable or hood connected

End of Section 6 -- Status
```

Instead of the first message, you could see:

```
Downloaded firmware currently running.
```

Instead of the last message you could see any one of the following:

```
Cable type connected: X.21 for X.27 male termination

Cable type connected: RS-232C/V.28 modem eliminator female termination

Cable type connected: V.35 male termination

Cable type connected: RS-366 with RS-232C male terminations

Cable type connected: RS-449 for RS-422/V.11 male termination (37 pin only)

Cable type connected: RS-232C/V.28 male termination

Cable type connected: Diagnostic hood
```

The HP-PB card provides additional information, as in the following example:

```
Max TRS registers = 1; Active TRS registers = 1

Firmware version number =

Card state: Selftest has completed.
```

Section 8 - Extended Hardware Test

Section 8 isolates those hardware errors that can be detected without requiring the user to manipulate the hardware. The security level is destructive.

Step 81 — Register Access Test: Exercises the card's slave circuitry and verifies that the hardware can write to and read from all 256 TRSs.

Step 82 — Memory test (Midbus = 13.5 sec.; HP-PB = 3.5 min.): Consists of a ROM checksum followed by a RAM test, exercising ROM and RAM bits and addressing. All of RAM is tested, so the TRS and most card state variables are overwritten when this command is executed. If this test completes, the card is left in something similar to the reset state.

Step 83 — Master circuitry test - Read16: Exercises the card master circuitry's Read16 capability.

Step 84 — Midbus master circuitry test - Chain execution: Tests the hardware required for normal chain execution and exercises the completion list and interrupt mechanism.

Quix test - tests the Quix's operation (this is step 84 for HP-PB cards).

Step 85 — (SRS CMD_STOP test): Checks the effects of a command stop on the 68000 microprocessor and the register save mechanism. This test uses downloaded test firmware to put the 68000 microprocessor into an infinite loop at non-maskable interrupt level 7 (i.e., nothing but a CMD_STOP or CMD_RESET can get it out of the loop).

Step 86 — Frontplane test: Checks the SCC chip, partially by looping 100 data bytes internally through the SCC at 64000 baud. Downloading firmware is required for this test also.

Step 87 — Selftest: This is the same test that Section 5 of the diagnostic consists of. It is included because it is the next logical test in this sequence.

Output:

```
Section 8 -- Internal Hardware

   Step 81 - Register Access Test

   Dots represent a successful write transaction to a TRS (slave)
   register set between 0 and 255.  If there was an error, an
   asterisk and an error message is displayed.
   ................................................................
   ................................................................
   ................................................................
   ................................................................
```

Dots represent a successful read transaction to a TRS (slave) register set between 0 and 255. If there was an error, an asterisk and an error message is displayed.

```
...............................................................
...............................................................
...............................................................

Register access test passed.

End of Step 81 - Register Access Test

End of Section 8 -- Internal Hardware

Section 8 -- Internal Hardware

   Step 82 - Card Memory Test

   Memory test passed.

   End of Step 82 - Card Memory Test

End of Section 8 -- Internal Hardware

Section 8 -- Internal Hardware

   Step 83 - Master Test: Read16

   Read16 test passed.

   End of Step 83 - Master Test: Read16

End of Section 8 -- Internal Hardware

Section 8 -- Internal Hardware

   Step 84 - Master Test: Chain Execution

   Chain execution test passed.
```

Note At this point, if you have an HP-PB system, the message displayed will be "Quix test passed" instead of "Chain execution test passed."

```
   End of Step 84 - Master Test: Chain Execution

End of Section 8 -- Internal Hardware

Section 8 -- Internal Hardware

   Step 85 - CMD_STOP Test
```

```
Downloaded firmware currently running.
Current firmware part number = 1818-xxxx; date code = 2808
Current firmware version: model = $6; revision = $0; option = $2
Cable type connected: X.21 for X.27 male termination

   CMD_STOP test passed.

   End of Step 85 - CMD_STOP Test

End of Section 8 -- Internal Hardware

Section 8 -- Internal Hardware

   Step 86 - Frontplane Test

Downloaded firmware currently running.
Current firmware part number = 1818-xxxx; date code = 2808
Current firmware version: model = $6; revision = $0; option = $2
Cable type connected: X.21 for X.27 male termination

   Frontplane test passed.

   End of Step 86 - Frontplane Test

End of Section 8 -- Internal Hardware

Section 8 -- Internal Hardware

   Step 87 - Selftest

   Selftest passed.

   End of Step 87 - Selftest

End of Section 8 -- Internal Hardware
```

Note

For HP-PB systems, Step 87 (Selftest) generates additional text. This HP-PB version follows immediately.

```
Section 8 -- Internal Hardware

   Step 87 - Selftest

   Performing a reset_HT...

   Performing a reset...

   Performing reads and writes to PSI card...
```

```
    Performing a selftest...

    Selftest passed.

    End of Step 87 - Selftest

End of Section 8 -- Internal Hardware
```

Section 9 - External Loopback

Section 9 tests the frontplane transceivers, cable and modem. The security level is destructive. Before executing the test, the user should attach an appropriate cable(s), test hoods, and modem if desired. The modem is set to loopback mode by hand, because the various configurations of modems make this difficult to do programmatically.

Step 91 Data Loopback - Non-interactive is the default step for Section 9. Uses the following loopback test parameters: 100 data bytes, baud rate 64000, cable type depends on what the firmware sees, 100 loopback.

Step 92 Data Loopback - Interactive prompts the user for byte count, baud rate, cable type to simulate, and repeat count. If the user enters (Return) to the prompts, the default values from Step 91 are used.

Output:

The numbers are for example only. Cable type will vary. If the user presses (Return) to the "Cable type to simulate" prompt, the default cable type is selected for him.

```
Section 9 -- External Loopback

Downloaded firmware currently running.
Current firmware part number = 1818-xxxx; date code = 2808
Current firmware model = $45; revision = $0; option = $2
Cable type connected: X.21 for X.27 male termination

   Step 91 - Data Loopback - Non-interactive

   Data loopback test passed.

   End of Step 91 - Data Loopback - Non-interactive

End of Section 9 -- External Loopback

Section 9 -- External Loopback

Downloaded firmware currently running.
Current firmware part number = 1818-xxxx; date code = 2808
Current firmware model = $45; revision = $0; option = $2
Cable type connected: X.21 for X.27 male termination
```

```
Step 92 - Data Loopback - Interactive

   Byte count (dec 2..4016, use '$' if hex) [4] =>26


   Legal baud rates
   ----------------
   300            2400            19200
   600            3600            38400
   1200           4800            56000
   1800           7200            64000
   2000           9600            120000

   Minimum baud rate = 300, maximum baud rate = 120000
   Type in an integer baud rate [64000] =>0

   LEGAL cable types to simulate
   -----------------------------
   0   No cable or hood connected - internal loopback
   1   X.21 for X.27 male termination
   2   RS-232C/V.28 modem eliminator female termination
   3   V.35 male termination
   4   RS-366 with RS-232C male terminations
   5   RS-449 for RS-422/V.11 male termination (37 pin only)
   6   RS-232C/V.28 male termination
   7   Diagnostic hood

   The default cable type is 1.

   Cable type to simulate =>0
   Repeat count (a 32 bit number, use '$' if hex) [1] =>2

   Dots represent a successful data loopback.  If There was
   an error, an asterisk and an error message is displayed.
   ..

   Data loopback test passed.

   End of Step 92 - Data Loopback - User Interactive

End of Section 9 -- External Loopback
```

Section 10 - Manufacturing Utilities

Section 10 provides the user with various tools for firmware and hardware testing. It is meant for manufacturing and should be very cautiously approached, as it can be executed while the data communication link is up. The security level is normal.

Step 101	CMD_STOP: This will result in the firmware saving the current contents of its processor registers, and possibly some other state information, into a reserved area of RAM, and then returning to ROM control.
Step 102	Peek: This step allows the user to look at RAM and ROM locations on the board. The user is prompted for an address or an I/O register number.
Step 103	Poke: This step allows the user to poke (insert new values into) RAM locations. The user is prompted for an address or an I/O register number.
Step 104	Start Microprocessor: This step allows the user to specify an address in the 68000 address space which contains the next instruction to be executed.
Step 105	Download: The user is prompted for the name of the file to be downloaded.
Step 106	Dump: RAM dump into a preallocated file.
Step 107	Card Status: Reads and displays card status information to verify firmware currently running on the card.
Step 108	Diagnostic Test Program (DTP)

Output:

The following output includes examples of selecting different steps from the PSIDAD menu.

```
Section 10 -- Manufacturing Utilities

   End of Step 101 - CMD_STOP

   101 - CMD_STOP   104 - START MICRO   107 - CARD STATUS
   102 - PEEK       105 - DOWNLOAD      108 - DIAGNOSTIC TEST PROGRAM
   103 - POKE       106 - DUMP          e,exit,<return> - EXIT
   Step number => 101

   Step 101 - CMD_STOP

   End of Step 101 - CMD_STOP
```

```
101 - CMD_STOP    104 - START MICRO    107 - CARD STATUS
102 - PEEK        105 - DOWNLOAD       108 - DIAGNOSTIC TEST PROGRAM
103 - POKE        106 - DUMP           e,exit,<return> - EXIT
Step number => 102

Step 102 - Peek

Hex starting address (no '$') =>FFFC0000
Byte count (a 32 bit number - use '$' if hex) [2] =>40

ADDRESS    DATA                                      ASCII

$FFFC0000  2222 2222 2222 2222  2222 2222 2222 2222
$FFFC0010  2222 2222 2222 2222  2222 2222 2222 2222
$FFFC0020  2222 2222 2222 2222

End of Step 102 - Peek

Step 102 - Peek

Hex starting address (no '$') =>FFFFFFFF
Byte count (a 32 bit number - use '$' if hex) [2] =>3

ADDRESS    DATA                                      ASCII

$FFFFFFFE  3333 3333                                 3333

End of Step 102 - Peek
Step 103 - Poke

Byte or word poke (0,1) [0] =>
Hex starting address (no '$') =>ffffc00
Hex byte =>12
Hex byte =>

End of Step 103 - Poke
```

```
101 - CMD_STOP   104 - START MICRO   107 - CARD STATUS
102 - PEEK       105 - DOWNLOAD      108 - DIAGNOSTIC TEST PROGRAM
103 - POKE       106 - DUMP          e,exit,<return> - EXIT
Step number =>103

Step 103 - Poke

Byte or word poke (0,1) [0] =>1
Even starting address (no '$') =>03
This is not a legal address.  Please try again.
Even starting address (no '$') =>04
Hex word =>34
Hex word =>

End of Step 103 - Poke

101 - CMD_STOP   104 - START MICRO   107 - CARD STATUS
102 - PEEK       105 - DOWNLOAD      108 - DIAGNOSTIC TEST PROGRAM
103 - POKE       106 - DUMP          e,exit,<return> - EXIT
Step number =>104

Step 104 - Start Microprocessor

Even starting address (no '$') =>FFFFFC15
Hex PSW (no '$') [2000] =>

End of Step 104 - Start Microprocessor
```

```
101 - CMD_STOP   104 - START MICRO   107 - CARD STATUS
102 - PEEK       105 - DOWNLOAD      108 - DIAGNOSTIC TEST PROGRAM
103 - POKE       106 - DUMP          e,exit,<return> - EXIT
Step number =>105

Step 105 - Download

The download filename must be no more than 8 characters long.
Filename =>dnldfile

End of Step 105 - Download

101 - CMD_STOP   104 - START MICRO   107 - CARD STATUS
102 - PEEK       105 - DOWNLOAD      108 - DIAGNOSTIC TEST PROGRAM
103 - POKE       106 - DUMP          e,exit,<return> - EXIT
Step number =>106

Step 106 - Dump

End of Step 106 - Dump
```

```
   101 - CMD_STOP   104 - START MICRO   107 - CARD STATUS
   102 - PEEK       105 - DOWNLOAD      108 - DIAGNOSTIC TEST PROGRAM
   103 - POKE       106 - DUMP          e,exit,<return> - EXIT
   Step number =>107


   Step 107 - Card Status

ROM firmware currently running.
Current firmware part number = 1818-xxxx; date code = 2808
Current firmware model = $20; revision = $1; option = $1
Cable type connected: X.21 for X.27 male termination

   End of Step 107 - Card Status

   101 - CMD_STOP   104 - START MICRO   107 - CARD STATUS
   102 - PEEK       105 - DOWNLOAD      108 - DIAGNOSTIC TEST PROGRAM
   103 - POKE       106 - DUMP          e,exit,<return> - EXIT
   Step number =>Return
End of Section 10 -- Manufacturing Utilities
```

Section 15 - EEPROM Failure History (HP-PB only)

This section makes a call to the PSI card to read EEPROM and retrieve selftest failure history. The failure history information is static (the data remains in EEPROM even though power to the system is turned off) so it is important to note that the information displayed is from the LAST selftest failure. This information is never cleared out, only overwritten by the next failure.

Output:

```
***************************************************************
****                                                    ******
****        PSIDAD PSI Device Adapter Diagnostic        ******
****                                                    ******
****       (C) Copyright Hewlett-Packard Co. 1988       ******
****                All Rights Reserved                 ******
****                  Version  A.01.01                  ******
****         Message File Version  A.01.01              ******
****                                                    ******
***************************************************************

Welcome, Today is Wed, Apr 05, 1989 04:13:18 PM

    Section 15 -- EEPROM failure history

Section 15 - Failure History
     This section reads and displays the LAST selftest failure information
     that was stored in the EEPROM's.  Important: The failure has no date
     associated with it, therefore the reason for the selftest failure
     may not still exist.

**** ERROR - This is where selftest failure information will be displayed.

    End of Section 15 -- EEPROM failure history
```

Error Messages

This section gives a list of the error messages that are generated by PSIDAD. The general action associated with any PSIDAD error message is to replace the PSI Device Adapter Card, unless otherwise stated in the error message. The "!" will be replaced by a parameter when the message is actually displayed.

809 `*** ERROR -- TRS ILLEGAL CMD TEST FAILED.  (PSIDADERR 809)`

`***Read of the TRS IO_COMMAND register does not match the illegal`

`***test command written to TRS IO_COMMAND.`

CAUSE Probable PSI Card failure

ACTION Replace PSI Card and reexecute PSIDAD

---

810 `*** ERROR -- ILLEGAL WRITE TO TRS IO_COMMAND UNDETECTED.`

`(PSIDADERR 810)`

`***The illegal test command written to the TRS IO_COMMAND register`

`***was not detected - no errors reported in TRS IO_STATUS.`

CAUSE Probable PSI Card failure

ACTION Replace PSI Card and reexecute PSIDAD

---

811 `*** ERROR -- UNEXPECTED IO_DMA_LINK CONTENTS.  (PSIDADERR 811)`

`***The hardware did not latch bus address bit in the slave status`

`***register.  TRS IO_COMMAND write and read of the illegal test`

`***command changed the contents of IO_DMA_LINK.`

CAUSE Probable PSI Card failure

ACTION Replace PSI Card and reexecute PSIDAD

---

812 `*** ERROR -- TRS IO_COMMAND WRITE/READ FAILED.  (PSIDADERR 812)`

`***Write of CMD_CLEAR to TRS IO_COMMAND does not match read.`

CAUSE Probable PSI Card failure

ACTION Replace PSI Card and reexecute PSIDAD

813	`*** ERROR -- CMD_CLEAR FAILURE.  (PSIDADERR 813)`
	`***CMD_CLEAR did not clear the *hard error command* caused by writing`
	`***an illegal test command to the TRS IO_COMMAND register.`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

816	`*** ERROR -- TRS TEST FAILURE.  (PSIDADERR 816)`
	`***IO_DMA_LINK read does not match write for TRS !.`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

840	`*** ERROR -- CMD_STOP DID NOT SAVE STACK POINTER.  (PSIDADERR 840)`
	`***The stack pointer was not saved into the *save byte* in RAM upon`
	`***writing a cmd_stop to the card.`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

919 *** ERROR -- DATA LOOPBACK ERROR. (PSIDADERR 919)

```
***During repeat #!, byte #! in the received data buffer does
***not match the corresponding byte in the sent data buffer.  The
***sent byte contains the value !, and the received byte the value
***!.  If nothing is connected to the card's front edge, then
***replace the card.  Otherwise, replace the most recent card front
***edge addition (cable or hood, modem ... ).

      ***SCC channel A:   WR0  = !   WR1  = !   WR2  = !   WR3  = !
      ***                 WR4  = !   WR5  = !   WR6  = !   WR7  = !
      ***                 WR8  = !   WR9  = !   WR10 = !   WR11 = !
      ***                 WR12 = !   WR13 = !   WR14 = !   WR15 = !
      ***
      ***                 RR0  = !   RR1  = !   RR2  = !   RR3  = !
      ***                 RR10 = !   RR12 = !   RR13 = !   RR15 = !
      ***
      ***SCC channel B:   WR0  = !   WR1  = !   WR2  = !   WR3  = !
      ***                 WR4  = !   WR5  = !   WR6  = !   WR7  = !
      ***                 WR8  = !   WR9  = !   WR10 = !   WR11 = !
      ***                 WR12 = !   WR13 = !   WR14 = !   WR15 = !
      ***
      ***                 RR0  = !   RR1  = !   RR2  = !   RR3  = !
      ***                 RR10 = !   RR12 = !   RR13 = !   RR15 = !
      ***
      ***DMA0 WRITE
      ***     status = !;  DMA0 error = !;  DMA0 transfer count = !;
      ***     device control = !;           operation control = !;
      ***     sequence control = !;         channel control = !;
      ***     mem addr register = !;  dev addr register = !;
      ***
      ***DMA0 READ
      ***     status = !;  DMA0 error = !;  DMA0 transfer count = !;
      ***     device control = !;           operation control = !;
      ***     sequence control = !;         channel control = !;
      ***     mem addr register = !;  dev addr register = !;
```

CAUSE Probable PSI Card or cable failure

ACTION Further loopback testing needed to determine PSI Card or cable failure.

5000	`*** ERROR -- PSI ALREADY IN USE BY DIAGNOSTIC SYSTEM. (PSIDADERR 5000)`
	`***Someone is already diagnosing the PSI that you requested.`
	`***It is illegal to have two copies of PSIDAD diagnosing the`
	`***same PSI at the same time.`

CAUSE	PSI Card already in use
ACTION	Make sure PSI Card is not already in use and reexecute `PSIDAD`

5011	`*** ERROR -- PROGRAM SERVICE CALL FAILED. (PSIDADERR 5011)` `***Program Service ! failed.`
CAUSE	Probable `PSIDAD` internal error
ACTION	Contact HP Support Personnel

5044	`*** ERROR -- ! IO_STATUS READY BIT NOT SET. (PSIDADERR 5044)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute `PSIDAD`

Error Message Headers

The messages listed in the following section are always displayed with either one or two more messages that tell you more about the error. Even though two or three error messages are displayed, only one error has been encountered by PSIDAD. The general action associated with any PSIDAD error message is to replace the PSI Device Adapter Card, unless otherwise stated in the succeeding error messages. The "!" will be replaced by a parameter when the message is actually displayed.

201	`*** ERROR -- RESET CALL FAILED  (PSIDADERR 201)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

202	`*** ERROR -- RESET FAILED  (PSIDADERR 202)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

315	`*** ERROR -- GET IO_DC CALL FAILED.  (PSIDADERR 315)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

316	`*** ERROR -- DAR VERSION CALL FAILED.  (PSIDADERR 316)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

501	`*** ERROR -- SELFTEST CALL FAILED.  (PSIDADERR 501)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

502	`*** ERROR -- SELFTEST FAILED.  (PSIDADERR 502)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

615	`*** ERROR -- READ CARD STATUS CALL FAILED.  (PSIDADERR 615)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

616	`*** ERROR -- READ CARD STATUS FAILED.  (PSIDADERR 616)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

804	`*** ERROR -- WRITE TO TRS IO_DMA_LINK REGISTER CALL FAILED. (PSIDADERR 804)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

805	`*** ERROR -- READ FROM TRS IO_DMA_LINK REGISTER CALL FAILED. (PSIDADERR 805)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

806	`*** ERROR -- IO_DMA_LINK READ/WRITE FAILED.  (PSIDADERR 806)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

807	`*** ERROR -- TRS IO_COMMAND REGISTER WRITE CALL FAILED.` `(PSIDADERR 807)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

808	`*** ERROR -- TRS IO_COMMAND REGISTER READ CALL FAILED.` `(PSIDADERR 808)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

811	`*** ERROR -- ILLEGAL WRITE TO TRS IO_COMMAND UNDETECTED.`
	`(PSIDADERR 811)`
	`***The illegal test command written to the TRS IO_COMMAND register`
	`***was not detected.  Instead, the following error was reported.`

CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

820	`*** ERROR -- MEMORY TEST CALL FAILED.  (PSIDADERR 820)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

821	`*** ERROR -- MEMORY TEST FAILED.  (PSIDADERR 821)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

824	`*** ERROR -- TRS IO_DMA_LINK WRITE FAILED.  (PSIDADERR 824)` `***Write physical address to TRS IO_DMA_LINK failed.`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

825	`*** ERROR -- CMD_RD16 FAILED.  (PSIDADERR 825)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

826	`*** ERROR -- READ16 FAILED.  (PSIDADERR 826)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

827	`*** ERROR -- READ FROM TRS IO_DMA_COMMAND REGISTER CALL FAILED.` `(PSIDADERR 827)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

828	***** ERROR -- READ FROM TRS IO_DMA_ADDRESS REGISTER CALL FAILED. (PSIDADERR 828)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

829	***** ERROR -- READ FROM TRS IO_DMA_COUNT REGISTER CALL FAILED. (PSIDADERR 829)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

836	***** ERROR -- CCMD_IN_MASTERTEST CALL FAILED. (PSIDADERR 836)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

837	***** ERROR -- CCMD_IN_MASTERTEST FAILED. (PSIDADERR 837)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

841	***** ERROR -- CMD_STOPTEST CALL FAILED. (PSIDADERR 841)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1006	`*** ERROR -- CMD_STOP WRITE FAILED.  (PSIDADERR 1006)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1007	`*** ERROR -- CMD_STOP FAILED.  (PSIDADERR 1007).`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1015	`*** ERROR -- CCMD_CTRL_PEEKADDR CALL FAILED.  (PSIDADERR 1015)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1016	`*** ERROR -- CCMD_CTRL_PEEKADDR FAILED.  (PSIDADERR 1016)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1017	`*** ERROR -- PEEK CALL FAILED.  (PSIDADERR 1017)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1018	`*** ERROR -- PEEK FAILED.  (PSIDADERR 1018)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1026	`*** ERROR -- POKE CALL FAILED.  (PSIDADERR 1026)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1027	`*** ERROR -- POKE FAILED.  (PSIDADERR 1027)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1032	`*** ERROR -- START MICROPROCESSOR CALL FAILED.  (PSIDADERR 1032)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1033	`*** ERROR -- START MICROPROCESSOR FAILED.  (PSIDADERR 1033)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1036	***** ERROR -- DOWNLOAD CALL FAILED. (PSIDADERR 1036)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

1048	***** ERROR -- DUMP CALL FAILED. (PSIDADERR 1048)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

5001	***** ERROR -- PSI_DAR INITIALIZATION #1 FAILED. (PSIDADERR 5001)**
CAUSE	Probable **PSIDAD** internal error
ACTION	Contact HP Support Personnel

5002	***** ERROR -- PSI_DAR INITIALIZATION #2 FAILED. (PSIDADERR 5002)**
CAUSE	Probable **PSIDAD** internal error
ACTION	Contact HP Support Personnel

5012	***** ERROR -- IO_STATUS CALL FAILED. (PSIDADERR 5012)**
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

5013	`*** ERROR -- TRANSMIT CALL FAILED.  (PSIDADERR 5013)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

5014	`*** ERROR -- TRANSMIT FAILED.  (PSIDADERR 5014)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

5015	`*** ERROR -- RECEIVE CALL FAILED.  (PSIDADERR 5015)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

5016	`*** ERROR -- RECEIVE FAILED.  (PSIDADERR 5016)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

5017	`*** ERROR -- ERROR REPORTED IN TRS IO_STATUS.  (PSIDADERR 5017)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

5018	`*** ERROR -- ERROR REPORTED IN SRS IO_STATUS.  (PSIDADERR 5018)`
CAUSE	Probable PSI Card failure
ACTION	Replace PSI Card and reexecute **PSIDAD**

3

HP-FL Adapter Diagnostic

Introduction

The HP-FL Diagnostic (HPFLDIAG) tests the HP-FL Adapter card on any HP Precision Architecture RISC computer system which supports the Online Diagnostic subsystem. HPFLDIAG will:

- Verify the integrity of the data path through the HP-FL subsystem (via loopback operations)
- Identify the product types of the various hardware modules
- Clear the subsystem hardware and run internal diagnostics (selftest)
- Obtain and decode status from the HP-FL interface card

Minimum Configuration

HPFLDIAG is designed to test and verify the HP-FL Adapter card configured and installed on any PA-RISC computer with zero or more peripherals connected across the HP-FL interface. In addition, LLIO manager diagnostic support must be provided for the HP-FL subsystem.

Auto-Diagnostics

If the Low Level I/O system detects a catastrophic error related to the HP-FL subsystem, a request may be made to execute HPFLDIAG in auto-diagnostic mode. In this mode, section 11 will be executed to perform a comprehensive check-out of the subsystem. If all tests are successful, HPFLDIAG will inform the system that the hardware is ok to use. If not, it will instead inform the system that the hardware is unusable. If at all possible, HPFLDIAG will determine the suspected field replaceable unit(s) that is causing the problem and report it.

Defects and Enhancements

Submit defect reports and enhancement requests for this diagnostic through the STARS database referencing Product Number 30600-10008.

Operating Instructions

The HP-FL Diagnostic is accessed by the user via the Diagnostic User Interface.

Default Tests

If you do not specify sections and steps to be run, the following default sections and steps will be executed depending on the current mode of the system:

Section 10	All Modes
Section 11	Destructive Mode Only

RUN Command

To bring up the Online Diagnostic subsystem, enter the following command at the system prompt:

```
sysdiag
```

The diagnostic subsystem responds with the following prompt indicating that access has been granted to the user:

```
DUI >
```

Typing `HELP` causes a summary of the DUI function and its commands to appear on the screen.

Note The device to be tested must be powered up and on line. The physical device location (pdev) shown below matches the same device shown on the "typical A1002A" system configuration, described in the chapter on DUI. The pdev value entered must be correct for the system being tested.

For example, to run the diagnostic in an MPE/iX environment, you might enter:

```
DUI > RUN HPFLDIAG pdev=4.3 <RUN Command Options>
                            |          |
                            |    none required for
                            |    default test suite
                            |
                insert physical location of
                device adapter to be tested here;
                alternatively, for MPE/iX ,
                type the ldev number;
                or for HP-UX, type the devfile name;
                or for HP-UX in this example, type pdev=4.3.0
```

All parameters available in the RUN command are acceptable as parameters when running this diagnostic. However, the only required parameter is the physical path (pdev) of an HP-FL interface card to be tested.

When HPFLDIAG is run, a header will be displayed; then, the first operation that will be performed will be to verify that the I/O path to the selected device is functioning properly. This will be accomplished by calling the IO_PATH_TEST service which does a series of loopbacks to the modules on the I/O path that precede the selected HPFL interface card. If this service encounters any problems in the path, the following message will be displayed:

```
*** WARNING THE I/O PATH TO THE SELECTED CARD MAY NOT BE FUNCTIONING
    PROPERLY  (HPFLWARN ####).
```

If this occurs, the problem which prompted the user to execute this diagnostic is most likely in one of the modules (CIO card, cables, bus converter, etc.) that are in the path from the host to the HPFL interface card.

Whether or not IO_PATH_TEST reported an error, the diagnostic will .

Next, HPFLDIAG will perform a rollcall operation on the HPFL interface card to determine what peripherals are connected to the card. If the rollcall fails, the following warning will be displayed:

```
*** WARNING -- THE IDENTITY OF THE DRIVES CONNECTED TO THE LINK
    HPFL CONTROLLER COULD NOT BE OBTAINED. SECTIONS 10 AND/OR 11
    SHOULD BE RUN IN ORDER TO ISOLATE THE PROBLEM. (HPFLWARN 5027)
```

Note that this warning implies an actual failure on the rollcall operation as opposed to a successfull rollcall which determined that no drives are connected. If sections 10 or 11 were selected in this particular run, they will isolate the problem that is causing rollcall to fail. If either of those sections were not selected, the diagnostic should be run again if the user wishes to have the problem isolated.

If the rollcall was successful, HPFLDIAG will display the drives that were found to be connected to the HP-FL interface card:

The HP-FL Interface card indicates that the following flex drives are connected:

```
Drive (n)
Drive (n)
  .
  .
```

```
.
Drive (n)
```

If the rollcall was not successful, a message will be displayed indicating that fact.

Several of the tests available in HPFLDIAG can be directed to one or more HPFL controllers connected to the HPFL interface card (e.g., step 12, step 22, section 10, etc.). In order to determine which, if any, of the controllers should be tested, the user will be prompted to input the controller(s) to which these tests should be directed. In order to assist the user in this determination, The prompt that is displayed is as follows:

```
The following drive(s) may be selected for testing:


   DRIVE n - selects a single drive to test where n is the number of
             the drive ;

   LINK - selects the LINK drive (the drive which is physically
          connected to the HPFL interface card via the optical
          cable);

   ALL - selects ALL drives (the LINK drive and all others connected
         to it);

   NONE - selects no drive. Only the HP-FL Interface card will be
          tested;

   Please input your selection (<CR> for ALL drives) >>
```

If the user selects a particular drive to test (by entering "DRIVE n"), all tests which are directed at a target drive will be directed to that drive only, regardless of whether or not the result of the prior rollcall operation indicated that the drive is actually connected. The following message will be displayed:

```
Drive (n) will be tested as the target drive.
```

If the user selects the LINK drive, HPFLDIAG will determine which drive is the link drive via an identify command directed at the link controller. If the identification fails for some reason, the following warning will be displayed:

```
*** WARNING -- THE IDENTITY OF THE LINK HPFL CONTROLLER COULD
    NOT BE OBTAINED. SECTIONS 10 AND/OR 11 SHOULD BE RUN IN ORDER
    TO ISOLATE THE PROBLEM. MEANWHILE, ANY TESTS THAT REQUIRE A
    TARGET DRIVE WILL NOT BE PERFORMED.
```

If this occurs, some tests, such as step 6, will not execute because they perform an operation on a specific target drive number. As for the trouble-tree sections (10 and 11), the portions of the trouble-trees that test the drive controller will not be executed.

If the identify command succeeds, the following message will be displayed to tell the user which target drive will be used:

```
The link drive (number n) will be tested.
```

The diagnostic will then proceed just as if the user had entered the link drive number at the prompt.

If the user selects ALL drives, `HPFLDIAG` will test all drives which were found to be connected via the rollcall command that was previously issued. If the rollcall command had failed or found no drives to be connected, no errors will be generated at this point but any tests that require a target drive will not execute. In either case, the following message will be displayed:

```
All drives will be tested.
```

Finally, if the user selects the NONE option, only HP-FL interface card tests will be executed. The following message will be displayed:

```
No target drives will be tested. Only HP-FL Interface card tests
will be performed.
```

If this occurs, some tests, such as step 6, will not execute because they perform an operation on a specific target drive number. As for the trouble-tree sections (10 and 11), the portions of the trouble-trees that test drive controllers will not be executed.

At this point, the sections and steps specified by the user will be executed and the results output. If the user did not specify sections and steps to be run, the default sections will be executed. If the diagnostic is running in Non-Exclusive/Non-Destructive or Excusive/Non-Destructive mode, the default section will be 10. Otherwise, for Exclusive/Destructive mode, sections 10 and 11 will be executed.

If, at any time, the number of errors generated exceeds the limit specified by the user in the ERRCOUNT parameter of the run command, the following message will be output:

```
THE MAXIMUM NUMBER OF ERROR  MESSAGES HAS BEEN REACHED (HPFLERR ####)
```

The diagnostic will then terminate execution. If the ERRPAUSE parameter of the RUN command was assigned a value of "on", then this diagnostic will stop after each error is generated and ask the user if the test should continue:

```
Do you wish to continue (Y/N)[ Y ]?
```

If the response is "Y" (or (Return)), the test will be resumed (if possible), and if the response is "N", this diagnostic will terminate. If the sections and steps specified by the user were executed the number of times specified in the LOOP parameter of the run command without the number of errors exceeding the ERRNUM value, the diagnostic will terminate normally and the following message output:

```
HP-FL Diagnostic Exiting . . .
```

Upon termination of this diagnostic, control will return to the Diagnostic system.

Test Section Descriptions

HPFLDIAG has the following sections and steps available:

Section 2	Clear
Step 6	Configure Clear Target Drive
Step 7	Reset Clear Target Drive
Step 8	Reset Interface
Section 3	Identify
Step 12	Target HP-FL Controller Identify
Step 13	Link Controller Identify
Step 14	HP-FL Interface Identify
Section 4	Loopback
Step 20	HPFL Interface Loopback
Step 21	Link Device Loopback
Step 22	Target Device Loopback
Step 23	HP-FL Interface Internal Loopback
Step 24	HP-FL Interface External Loopback
Section 6	HP-FL Interface Global Status
Section 10	Verification Trouble Tree
Section 11	Diagnostic Trouble Tree
Section 13	F/W Upgrade (NIO HP-FL Only)

Section 2—CLEAR

This section provides the user with the means of performing several different types of clear operations on the HP-FL hardware. Destructive mode will be needed to execute this section.

Step 6 - Configure Clear	Issues a configure clear command to the target HPFL controller. If no target drive was selected for testing, this step will not execute and an error message will be displayed to the user. If all drives were selected for testing, the clear will be issued to each drive that is known to be connected. This is a soft clear which brings the device into a known state. No internal drive selftest will be run as a result of executing this command.
Step 7 - Reset Clear	Issues a reset clear command to the target HPFL controller. If no target drive was selected for testing, this step will not execute and an error message will be displayed to the user. If all drives were selected for testing, the clear will be issued to each drive that is known to be connected. This command will reset all device hardware and software just as if the power switch was cycled. Power-on selftest will be run on the drive and the results displayed if not successful.
Step 8 - Reset Interface	Issues a reset command to the selected HPFL interface card. Power-on selftest will be run on the HPFL interface card and the results displayed if not successful.

OUTPUT:

```
Section 2 -- Clear

   Clearing drive (n) ...
   .
   .
   .
   Clearing drive (n) ...


   Step 6 - Configure Clear Completed


   Resetting drive (n) ...
   .
   .
   .
   Resetting drive (n) ...


   Step 7 - Reset Clear Completed

   Step 8 - Reset Interface Completed


End of Section 2 -- Clear
```

ERROR MESSAGES:

```
*** THE SELECTED DEVICE FAILED ITS INTERNAL SELFTEST.  (DSSERR 539)

*** WARNING -- A TARGET DRIVE MUST HAVE BEEN SELECTED IN ORDER
    TO RUN THIS STEP. NO OPERATION WAS PERFORMED.  (HPFLWARN 5024)
```

Section 3—IDENTIFY

This section provides the user with the means of identifying the various hardware modules in the HP-FL subsystem. This section will run in any mode.

Step 12 - HPFL Controller Identify	Issues an identify command to the target HPFL controller, and then decodes and displays the returned information. If no target drive was selected for testing, this step will not execute and an error message will be displayed to the user. If all drives were selected for testing, the identify will be issued to each drive that is known to be connected.
Step 13 - Link Controller Identify	Issues an identify command to the well-known virtual circuit of the link HPFL controller (i.e., the controller connected directly to the host via a fiber optic cable), and then decodes and displays the returned information. If no drives were selected for testing, this step will not execute and an error message will be displayed to the user.
Step 14 - HPFL Interface Identify	Issues an identify command to the selected HPFL interface card, decodes and displays the returned information.

OUTPUT:

```
Section 3 -- Identify

   Identifying drive (n) ...

   Identity class -- HPFL host SPU interface
                                or
                     HPFL multiplexer
   Device class   -- Pseudo device
                           or
                     CS/80
                           or
                     Class unknown
   Deadlock avoidance scheme -- None (full duplex device)
                                          or
                                Half duplex master device
                                          or
                                Half duplex slave device
                                          or
                                Not defined
   Protocol controller revision code -- nn


   Identifying drive (n)

   (identity data for this drive)
   .
   .
   .

   Identifying drive (n)
```

```
(identity data for this drive)


Step 12 - HPFL Controller Identify Completed

Identity class -- HPFL host SPU interface
                            or
                  HPFL multiplexer
Device class   -- Pseudo device
                        or
                  CS/80
                        or
                  Class unknown
Deadlock avoidance scheme -- None (full duplex device)
                                       or
                             Half duplex master device
                                       or
                             Half duplex slave device
                                       or
                             Not defined
Protocol controller revision code -- nn
Number of receive errors since last heartbeat -- nn
Controller number -- 0


Step 13 - Link Controller Identify Completed

(CIO HP-FL)
        Card ID - NN
        Firmware ID - NN
        Firmware revision date - NNNN
        MPX - NN
        Hardware revision - NN
        Mode - NN
        Number of active ports - NN
        Number of requests per port - NN
        Data blocking factor - NNNN

(NIO HP-FL)
        Card hardware version model          -- $NN
                              revision       -- $NN
        Card software version model          -- $NN
                              revision       -- $NN
                              option         -- $NN
        Card IODC revision                   -- NN
        Card firmware datecode               -- NNNN
        Card EEPROM status                   -- failed erase
                            or
        Card EEPROM status                   -- erased
                            or
        Card EEPROM status                   -- valid, no updates
```

```
                         or
Card EEPROM status                          -- disabled
                         or
Card EEPROM status                          -- unknown: $NN
                         or
Card EEPROM status                          -- active with updates
Card EEPROM patch datecode                  -- NNNN
The link is up.
                         or
The link is sick.
                         or
The link is down - receive fiber not active.
                         or
The link is down - receive fiber intermittent.
                         or
The link is down - no response from link device.
                         or
The link is down - excess transmit data errors.
                         or
The link is down - excess receive data errors.
                         or
The link is down - link device resynchronized link.
                         or
The link is down - link device sent remote reset.
                         or
The link is down - HPFL Interface resynchronized link.
                         or
The link is down - no resynchronize response from link.
                         or
The link is down - improper configuration.
Card configuration is more equal


Step 14 - HPFL Interface Identify Completed


End of Section 3 -- Identify
```

ERROR MESSAGES:

```
*** WARNING -- A TARGET DRIVE MUST HAVE BEEN SELECTED IN ORDER
    TO RUN THIS STEP. NO OPERATION WAS PERFORMED. (HPFLWARN 5024)

*** WARNING -- TESTING OF THE HP-FL CARD ONLY WAS SELECTED.
    AT LEAST ONE DRIVE MUST HAVE BEEN SELECTED FOR TESTING IN
    ORDER TO RUN THIS STEP. NO OPERATION WAS PERFORMED.
    (HPFLWARN 5028)
```

Section 4—LOOPBACK

This section provides the user with access to various forms of loopback. The data pattern that will be used in each loopback operation will follow the pattern:

0,1, ... ,254,255.

This pattern will be repeated, if necessary, for the entire length of the loopback.

The pattern received from the device will be compared with the pattern sent to verify correct transmission. If the pattern was corrupted, the bytes in error will be displayed to the user.

Although a brief description of the paths covered by the loopback is given in each step, the HP-FL Diagnostics ERS should be consulted for more complete details. Refer to each step description for the test mode necessary to execute it.

Step 20 - HPFL Interface Loopback	Issues a CIO loopback to the selected HPFL interface card. This will verify that the data path across the backplane is working properly. The length of the loopback will be 256 bytes for CIO HP-FL and NIO HP-FL. This step will run in any mode.
Step 21 - Link Device Loopback	Issues a loopback command to the well known virtual circuit of the HPFL controller connected to the host (i.e., the link controller). This test will verify that the data path from the HPFL interface adapter RAM to the HPFL controller card on the link drive is working properly. The length of the loopback will be 256 bytes for CIO HP-FL and NIO HP-FL. This step will run in any mode.
Step 22 - Target Device Loopback	Issues a loopback command to the target HPFL controller. If no target drive was selected for testing, this step will not execute and an error message will be displayed to the user. If all drives were selected for testing, the loopback will be issued to each drive that is known to be connected. This test will verify that the data path from the HPFL Interface card RAM to the DMA on the specified HPFL controller card is working properly. The pattern for the loopback will be 32 K bytes in length. This step will run in any mode.
Step 23 - HPFL Interface Internal Loopback	Issues an internal loopback command to the selected HPFL interface card. This test is completely internal to the card. It verifies that the data path within the HPFL Interface card is working properly. This does not include the optical components of the card. The loopback pattern will be 256 bytes in length for CIO HP-FL, and 224 for NIO HP-FL. Destructive mode is required to execute this test.

Step 24 - HPFL Interface External Loopback

Issues an external loopback command to the selected HPFL interface card. This test covers the same paths that step 23 does with the addition of the optical components of the HPFL controller card. This loopback will be 256 bytes in length for CIO HP-FL, and 224 for NIO HP-FL, and requires that a loopback fiber be installed on the HPFL interface card. The diagnostic will verify whether or not a loopback fiber has been installed on the card. If not, the following warning will be displayed to the user:

```
*** WARNING -- A LOOPBACK FIBER MUST BE INSTALLED ON THE
HPFL INTERFACE CARD IN ORDER FOR THIS TEST TO BE VALID. THIS
CAN BE DONE BY CONNECTING THE TRANSMIT AND RECEIVE PORTS WITH
A SNGLE FIBER. IF YOU STILL WISH TO EXECUTE THIS TEST, PLEASE
INSTALL A FIBER AND THEN ANSWER "Y" TO THE FOLLOWING PROMPT.
IF YOU DO NOT WISH TO CONTINUE, ANSWER "N" TO THE PROMPT. YOU
WILL NOT BE ALLOWED TO PROCEED WITH THIS TEST UNTIL A FIBER IS
PROPERLY INSTALLED.

Do you wish to continue (Y/N)[N]?
```

The user is then expected to either terminate the test by responding "N" to the prompt, or to install a loopback fiber and then reply "Y" to the prompt. If a "Y" response is given, the diagnostic will again verify whether or not a fiber has been installed. If so, the test will continue. If not, the warning and prompt wll be re-displayed. Destructive mode is required to execute this test.

OUTPUT:

```
Section 4 -- Loopback

   Step 20 - CIO Loopback Completed


   Step 21 - Link Device Loopback Completed


   Initiating loopback to drive (n)
   .
   .
   .
   Initiating loopback to drive (n)


   Step 22 - Target Device Loopback Completed


   Step 23 - HPFL Interface Internal Loopback Completed


   Step 24 - HPFL Interface External Loopback Completed
```

End of Section 4 -- Loopback

ERROR MESSAGES:

```
*** ERROR IN TRANSMISSION DETECTED DURING LOOPBACK.  (HPFLERR 5009)

                                                  Bit Positions
            Hex Value          Hex Value          In Error
  Byte #    Transmitted        Received             01234567
  ======    ===========        ===========        =============
   12          2E                 2C                00000010
   33          57                 33                01100100
   .
   .
   .
   241         3C                 3A                00000110

     Note -- entries in the preceeding table will be printed
             for as many errors as were detected, unless the
             ERRNUM value is exceeded.


*** WARNING -- A TARGET DRIVE MUST HAVE BEEN SELECTED IN ORDER
   TO RUN THIS STEP. NO OPERATION WAS PERFORMED.  (HPFLWARN 5024)

*** WARNING -- TESTING OF THE HP-FL CARD ONLY WAS SELECTED.
    AT LEAST ONE DRIVE MUST HAVE BEEN SELECTED FOR TESTING IN ORDER
    TO RUN THIS STEP. NO OPERATION WAS PERFORMED.  (HPFLWARN 5028)
```

Section 6—STATUS

This section obtains and decodes hardware status from the selected HP-FL interface card.

OUTPUT:

```
Section 6 -- Status
   HPFL Global Status:
   (CIO HP-FL)
      Out of Lock  -- (true or false)
      Optical state -- (active or not active)
      More Equal -- (true or false)
      Link State -- (up or down)
      Jupiter Loopback Mode - (true or false)
      Raw Mode -- (true or false)
      Non-Maskable Interrupt -- (set or not set)
      Link (is or is not) dead or dying
      Activity -- (true or false)
      More Equal jumpers are correctly configured
                     or
      More Equal jumpers are mis-configured
      Link (is or is not) performing at normal expectation
      Link (is or is not) responding to requests
      Last Self-Test (passed or failed)
         {if last selftest failed:}
         Failure code -- nn
      Number of Cumulative Link Errors -- nn
      Elapsed Time since last reset -- nn seconds
      nn errors have been detected in the last hour

   (NIO HP-FL)
      Asynchronous event:  Link down - receive fiber not active
          Current recovered data error count:          NN
          Current count period:                        NN seconds
          Total data errors recovered by HPFL Interface:  NN
          Total data errors recovered by link device:     NN
          Total time since last Interface reset:       NN seconds
                     or
      Asynchronous event:  Link down - receive fiber intermittent
          Current recovered data error count:          NN
          Current count period:                        NN seconds
          Total data errors recovered by HPFL Interface:  NN
          Total data errors recovered by link device:     NN
          Total time since last Interface reset:       NN seconds
                     or
      Asynchronous event:  Link down - no response from link device
          Current recovered data error count:          NN
          Current count period:                        NN seconds
          Total data errors recovered by HPFL Interface:  NN
          Total data errors recovered by link device:     NN
          Total time since last Interface reset:       NN seconds
```

```
            or
Asynchronous event:  Link down - excess transmit data errors
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
            or
Asynchronous event:  Link down - excess receive data errors
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
            or
Asynchronous event:  Link down - link device resynchronized link
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
            or
Asynchronous event:  Link down - link device sent remote reset
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
            or
Asynchronous event:  Link down - HPFL Interface resynchronized link
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
            or
Asynchronous event:  Link down - no resynchronize response from link
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
            or
Asynchronous event:  Link down - improper configuration
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
            or
Asynchronous event:  Link sick - significant transmit data errors
```

```
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
                or
Asynchronous event:  Link sick - significant receive data errors
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
                or
Asynchronous event:  Recovered link error - transmit data
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
                or
Asynchronous event:  Recovered link error - receive data
    Current recovered data error count:            NN
    Current count period:                          NN seconds
    Total data errors recovered by HPFL Interface: NN
    Total data errors recovered by link device:    NN
    Total time since last Interface reset:         NN seconds
                or
Asynchronous event:  Link up
                or
Asynchronous event:  Remote link reset
                or
Asynchronous event:  Unexpected message
                or
Asynchronous event:  Unexpected request to send message
                or
Asynchronous event:  Unexpected ready to send message
                or
Asynchronous event:  Unexpected ready to receive message
                or
Asynchronous event:  Immediate status from drive
                or
Asynchronous event:  Unknown - O$NN
                or
Transaction error:  Aborted by host
Transaction error:  Aborted by device
Transaction error:  Interface command error - invalid command
Transaction error:  Interface command error - invalid sub-command
Transaction error:  Interface command error - invalid address/counts
Transaction error:  Interface command error - missing device command
Transaction error:  Interface command error - mismatched loopback
Transaction error:  Interface command error - missing loopback read
```

```
Transaction error:  Interface command error - invalid loopback size
Transaction error:  Interface command error - invalid bank
Transaction error:  Aborted by link resynchronization
Transaction error:  Aborted by remote link reset
Transaction error:  Interface command error - invalid TEB size
Transaction error:  Interface command error - data length too small
Transaction error:  Interface command error - HP-PB loopback size
Transaction error:  Interface unable to enter internal loopback mode
Transaction error:  Interface disabled by selftest fault
Transaction error:  Interface unable to start command - link down
Transaction error:  Interface command error - too much device status
Transaction error:  Interface error - out of resources
Transaction error:  Interface error - level 3 protocol error
Transaction error:  Interface command error - EEPROM active
Transaction error:  Interface unable to program EEPROM
Transaction error:  Interface unable to erase EEPROM
Transaction error:  Unknown - O$NN
Auxiliary transaction status:
    O$NN  O$NN  O$NN  O$NN  O$NN
HPFL Interface operational
HPFL Interface not ready
HPFL Interface reports fatal error:
    Fiber link protocol controller failure
    HP-PB parity error
    EEPROM failure
    Invalid SRS command received
    HP-PB timeout error
    Assertion of HP-PB PATH_ERROR
    HP-PB protocol error
    HP-PB non-responding address
    Selftest detected H/W failure
    Interface configuration invalid
    Unknown error - O$NN
Unknown status format (O$NN) - status dump:
       O$NN  O$NN  O$NN  O$NN  O$NN


End of Section 6 -- Status
```

Section 10—VERIFICATION TROUBLE TREE

This section is designed to verify that the communication path from the host to the drive cluster is functioning properly. If not, the suspected causes of the problem will be reported. Several possible output scenarios from this section are given below. Note that this is not an exhaustive set of possibilities but is provided to give the user an idea of the type of output this section will produce. This section can be run in any mode.

OUTPUT:

```
Section 10 -- Verification Trouble Tree

Scenario 1:

   Local loopback to the HPFL interface card failed.
      Suspected failing FRU(s) are (in order of probability):
         HPFL interface card.
         CIO channel adapter. (* only displayed for CIO FL *)


Scenario 2:

   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller passed.
   Identification of HPFL controller (n) passed.
   Loopback to HPFL controller (n) failed.
      Suspected failing FRU(s) are (in order of probability):
         HPFL controller (n).

   Identification of HPFL controller (n) passed.
   .
   .
   .


Scenario 3:

   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller passed.
   Identification of HPFL controller (n) passed.
   Loopback to HPFL controller (n) passed.

   No problems have been detected in the sub-system from the host
   to the HPFL controller(n). If you suspect that there may be
   problems with this drive, run FLEXDIAG on the drive.

   Identification of HPFL controller (n) passed.
   .
   .
   .
```

```
Scenario 4:

   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller passed.
   Identification of HPFL controller (n) failed.
      Suspected failing FRU(s) are (in order of probability):
         Target drive (n) may be powered down.
         Optical cable may be damaged or disconnected.
         Target drive (n) may be at the wrong address.
         Pbus cables may be damaged or connections are bad.

   More information, and perhaps further isolation of the problem,
   may be obtained by following the recommended further action
   given below.

      Recommended Further Action
         Run On-Site section (12) on the link drive

   Identification of HPFL controller (n) passed.
   .
   .
   .


Scenario 5:

   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller failed.
      Suspected failing FRU(s) are (in order of probability):
         Optical cable may be damaged or disconnected.
         Link HPFL controller.
         HPFL interface card.

   More information, and perhaps further isolation of the problem,
   may be obtained by following the recommended further action
   given below.

      Recommended Further Action
         Run On-Site section (12) on the link drive


End of Section 10 -- Verification Trouble Tree
```

Section 11—DIAGNOSTIC TROUBLE TREE

This section is designed to verify that the HP-FL subsystem is functioning correctly. This section is more exhaustive than section 10 due to the fact that this section runs in Destructive mode. Several possible output scenarios from this section are given below. Note that this is not an exhaustive set of possibilities but is provided to give the user an idea of the type of output this section will produce.

OUTPUT:

```
Section 11 -- Diagnostic Trouble Tree

Scenario 1:

   Reset of the HPFL interface card failed.
      Suspected failing FRU(s) are (in order of probability):
         HPFL interface card
         CIO channel adapter. (* only displayed for CIO FL *)


Scenario 2:

   Reset of the HPFL interface card passed.
   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller passed.
   Identification of HPFL controller (n) passed.
   Reset clear of HPFL controller (n) passed.
   Configure clear of HPFL controller (n) passed.
   Loopback to HPFL controller (n) failed.
      Suspected failing FRU(s) are (in order of probability):
         HPFL controller (n).

   Identification of HPFL controller (n) passed.
   .
   .
   .



Scenario 3:

   Reset of the HPFL interface card passed.
   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller passed.
   Identification of HPFL controller (n) passed.
   Reset clear of HPFL controller (n) failed.

      Suspected failing FRU(s) are (in order of probability):
         HPFL controller (n).

   Identification of HPFL controller (n) passed.
```

```
 .
 .
 .


Scenario 4:
   Reset of the HPFL interface card passed.
   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller passed.
   Identification of HPFL controller (n) failed.
      Suspected failing FRU(s) are (in order of probability):
         Target drive (n) may be powered down.
         Optical cable may be damaged or disconnected.
         Target drive (n) may be at the wrong address.
         Pbus cables may be damaged or connections are bad.

   More information, and perhaps further isolation of the problem,
   may be obtained by following the recommended further action
   given below.

      Recommended Further Action
         Run On-Site section (12) on the link drive

   Identification of HPFL controller (n) passed.
   .
   .
   .


Scenario 5:

   Reset of the HPFL interface card passed.
   CIO loopback to the HPFL interface card passed.
   Loopback to the link HPFL controller passed.
   Identification of HPFL controller (n) passed.
   Reset clear of HPFL controller (n) passed.
   Configure clear of HPFL controller (n) failed.
      Suspected failing FRU(s) are (in order of probability):
         HPFL controller (n).

   Identification of HPFL controller (n) passed.
   .
   .
   .

End of Section 11 -- Diagnostic Trouble Tree
```

Section 13—F/W UPGRADE (NIO HP-FL Only)

This section provides the capability to upgrade NIO HP-FL device adapter firmware without the usual ROM swap. The user will be asked to provide a fully qualified filename for a F/W patch file. This patch file must be compatible with the ROMs on the device adapter (i.e., the ROM datecode in the patch file must match the ROM datecode of the device adapter). If the datecodes do not match an error message is generated and no update is done.

OUTPUT:

```
Section 13 -- F/W Upgrade

*** WARNING -- THIS SECTION WILL ERASE THE HPFL INTERFACE'S
EEPROM AND REPROGRAM IT WITH NEW FIRMWARE UPDATES.  THIS PROCESS
WILL REQUIRE THAT YOU HAVE OBTAINED A SUITABLE UPDATE FILE
AND HAVE IT PROPERLY INSTALLED ON THE SYSTEM.

Do you wish to continue (Yes/No)[No]?

Please input fully qualified firmware upgrade filename
(<CR> to exit upgrade) >>

Current HPFL Interface F/W revision and upgrade status:
   Card firmware datecode                  -- NNNN
   Card EEPROM status                      -- active with updates
   Card EEPROM patch datecode              -- NNNN
   Card EEPROM erase count                 -- NN
Selected HPFL Interface F/W upgrade:
   Card firmware datecode                  -- NNNN
   Card EEPROM patch datecode              -- NNNN
           or
   Null patch selected.
   Firmware in EEPROMS will be erased.
Do you wish to continue (Yes/No)[No]?

HPFLDIAG building new EEPROM image -- Please wait . . . .

HPFLDIAG downloading to HPFL Interface -- Please wait . . . .

HPFL Interface firmware successfully upgraded.
HPFL Interface firmware updates in EEPROM now in effect.
           or
HPFL Interface EEPROM successfully erased.
HPFL Interface executing firmware only in ROM.


End of Section 13 -- F/W Upgrade
```

ERROR MESSAGES:

```
*** FIRMWARE UPGRADE NOT SUPPORTED BY HPnnnnnA.  (HPFLERR 5032)

*** FILE: ! NOT FOUND.  (HPFLERR 5033)

*** AN ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO OPEN A FILE
    (HPFLERR 5034)

*** AN ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO READ A FILE
    (HPFLERR 5034)

*** FIRMWARE UPGRADE FILE [!] IS INVALID.  (HPFLERR 5036)

*** FIRMWARE AND UPGRADE FILE VERSIONS DO NOT MATCH.  (HPFLERR 5037)

*** FIRMWARE UPGRADE FAILURE.  THE HPFL INTERFACE IS NOW
    FUNCTIONING ON A NON-CURRENT FIRMWARE AND/OR EEPROM PATCH
    REVISION.  (HPFLERR 5038)
```

Error Messages

This section gives a complete list of the error messages that may be generated by `HPFLDIAG` along with brief explanations of the meaning of the messages. The messages will be listed in numerical order and are exactly as they appear in the message catalog. The "!" indicates that a parameter of some sort will replace the exclamation point when the message is displayed.

5000	**`*** WARNING -- THE I/O PATH TO THE SELECTED CARD MAY NOT BE FUNCTIONING PROPERLY (HPFLWARN 5000)`**
CAUSE	An error was detected by the Io_Path_Test service while testing the modules on the i/o path preceding the selected device.
ACTION	Execute the appropriate diagnostics on the modules preceding the selected device on the i/o path, especially on those that may have been reported as faulty in error messages immediately preceding this message. Note that the results of the execution of this instance of `HPFLDIAG` may be invalid.
5001	**`*** DEVICE FAILED TO RESPOND TO ! COMMAND (HPFLERR 5001)`**
CAUSE	No response to an i/o was received prior to the expiration of the allotted time.
ACTION	Verify that the selected device is actually connected to the system. Run SYSMAP, if available, to confirm the presence of the device.
5002	**`*** HPFL DIAGNOSTIC TERMINATING (HPFLERR 5002)`**
CAUSE	A fatal error has been encountered.
ACTION	The specific error that was encountered should have been reported immediately prior to this message. Follow the action instructions for that error message.
5003	**`*** ! COMMAND IS NOT IMPLEMENTED ON THIS DRIVE/SYSTEM (HPFLERR 5003)`**
CAUSE	The selected operation is either not implemented on the selected device or the system does not provide access to it.
ACTION	This operation is unavailable.
5004	**`*** DEVICE ENCOUNTERED AN ERROR WHILE EXECUTING AN ! COMMAND (HPFLERR 5004)`**
CAUSE	The device reported an error as a result of executing the selected operation.
ACTION	Most likely, a hardware problem exists in the sub-system. Run the trouble-tree sections of this diagnostic to isolate the failing FRU.

5005 ***** THE MAXIMUM NUMBER OF ERRORS HAS BEEN EXCEEDED (HPFLERR 5005)**

CAUSE The user specified error limit has been reached.

ACTION If more errors are desired, rerun the diagnostic assigning a larger value to the ERRCOUNT parameter of the run command.

5006 ***** AN UNRECOGNIZED REPLY WAS FOUND (HPFLERR 5006)**

CAUSE The reply that was entered in response to a prompt by the diagnostic is not valid.

ACTION Refer to the prompt that was displayed and enter a response that is within the specified list of valid responses.

5007 ***** A NUMERICAL INPUT WAS EXPECTED BUT NOT RECEIVED (HPFLERR 5007)**

CAUSE The reply that was entered in response to a prompt by the diagnostic is not a valid number.

ACTION Reenter number using only numeric characters and valid special characters (e.g. +, -, , etc.).

5008 ***** AN UNEXPECTED ERROR OCCURRED WHILE ATTEMPTING TO COMMUNICATE WITH THE DEVICE. (HPFLERR 5008)**

CAUSE A call to the HPFL device access routine resulted in an unexpected status return.

ACTION The specific status generated by the DAR should have been displayed immediately prior to this error message - refer to the cause-action text for that message for more information. This is possibly caused by a mis-match of the driver and diagnostic system software. Ensure that the diagnostic system currently installed is the correct one and, if so, report this problem via an SR.

5009 ***** ERROR IN TRANSMISSION DETECTED DURING& READ LOOPBACK TEST: (HPFLERR 5009)**

```
                                                     Bit Positions
                    Octal Value       Octal Value    In Error
        Byte #      Transmitted       Received        01234567
        ======      ===========       ===========    ============
```

CAUSE One or more bytes of data that were received in a loopback operation did not contain the expected value(s). Data is being corrupted along the data path either between the host and the HPFL interface card or within the HPFL sub-system itself.

ACTION If this error is generated within a trouble-tree section, follow the directions specified to isolate the most suspect failing FRU(s). If not, execute the trouble-tree sections of this diagnostic to isolate the failing FRU(s).

5010	! ! ! !
CAUSE	This message is generated in conjunction with one or more other error messages and is used for data formatting.
ACTION	Refer to the accompanying message(s) for appropriate cause-action text.
5011	***** ERROR -- EXPECTED ! BYTES FROM THE DEVICE AND RECEIVED ! BYTES (HPFLERR 5011)**
CAUSE	The number of bytes in the reply from the device was not what was expected. This is most likely a result of executing the diagnostic on a drive which is not supported by it.
ACTION	Verify that the selected device is in the list of supported devices for the diagnostic (LIST ALL from the DUI). If it is, execute the trouble-tree sections of this diagnostic and follow the directions to isolate the failing FRU(s).
5013	***** NO OPERATION WAS PERFORMED (HPFLERR 5013)**
CAUSE	Due to a previous error, which has already been reported, no operation was performed.
ACTION	Refer to action instructions for previously reported error.
5014	***** AN ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO SEND/RECEIVE INFORMATION FROM THE USER (HPFLERR 5014)**
CAUSE	Due to a previous error, which has already been reported, the diagnostic was unable to communicate with the user interface process.
ACTION	Refer to action instructions for previously reported error.
5015	***** AN ERROR WAS ENCOUNTERED IN ATTEMPTING TO RETRIEVE A MESSAGE FROM THE CATALOG (HPFLERR 5015)**
CAUSE	An error was returned while attempting to obtain a message from the catalog. The actual error will have been displayed prior to this message.
ACTION	This is a software error. Most likely, the message catalog and the diagnostic code are mis-matched. If possible, obtain the correct diagnostic and catalog source and re-install it. This problem should be reported via an SR.

5016	***** AN ERROR WAS ENCOUNTERED IN ATTEMPTING TO CONVERT A NUMBER TO A STRING (HPFLERR 5016)**
CAUSE	Due to a previous error, which has already been reported, the diagnostic was unable to convert a number to a string.
ACTION	Refer to action instructions for previously reported error.

5017	***** AN ERROR WAS ENCOUNTERED IN ATTEMPTING A BIT EXTRACTION OPERATION (HPFLERR 5017)**
CAUSE	Due to a previous error, which has already been reported, the diagnostic was unable to extract one or more bits from a number.
ACTION	Refer to action instructions for previously reported error.

5018	***** THE SELECTED DEVICE COULD NOT BE OBTAINED FOR TESTING (HPFLERR 5018)**
CAUSE	Access to the selected device was not granted by the diagnostic system. The particular reasons for this should have been displayed prior to this error message.
ACTION	Refer to instructions for previously reported errors.

5019	***** YOUR RESPONSE WAS INVALID (HPFLERR 5019)**
CAUSE	The data entered in response to a prompt was not valid.
ACTION	Refer to the prompt to determine the valid responses for the particular situation and enter one of the specified valid responses.

5020	***** AN ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO OBTAIN DATA FROM AN I/O BUFFER (HPFLERR 5020)**
CAUSE	Due to a previous error, which has already been reported, the diagnostic was unable to get data from its i/o buffer and, therefore cannot obtain data from the device.
ACTION	Refer to action instructions for previously reported error.

5021 ***** AN ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO PLACE DATA INTO AN I/O BUFFER (HPFLERR 5021)**

CAUSE Due to a previous error, which has already been reported, the diagnostic was unable place data into its i/o buffer and, therefore, cannot send data to the device.

ACTION Refer to action instructions for previously reported error.

5022 ***** AN ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO OBTAIN AN I/O BUFFER (HPFLERR 5022)**

CAUSE Due to a previous error, which has already been reported, the diagnostic was unable obtain an i/o buffer and therefore, cannot send/receive data to/from the device.

ACTION Refer to action instructions for previously reported error.

5023 ***** AN ERROR OCCURRED WHILE ATTEMPTING TO INFORM THE SYSTEM, THAT THE DEVICE IS BROKEN. (HPFLERR 5023)**

CAUSE Due to a previous error, which has already been reported, the diagnostic was unable inform the diagnostic system that the selected device is broken.

ACTION Refer to action instructions for previously reported error. Also, if this error was not generated within a trouble-tree section, execute the trouble-tree sections to isolate the failing FRU(s).

5024 ***** WARNING -- A TARGET DRIVE MUST HAVE BEEN SELECTED IN ORDER TO RUN THIS STEP. NO OPERATION WAS PERFORMED. (HPFLWARN 5024)**

CAUSE A step was selected that requires a target drive number in order to execute.

ACTION Rerun the step, this time selecting a target drive when the target drive prompt is given during initialization.

5025 ***** THE SELECTED TARGET DRIVE IS NOT VALID. (HPFLERR 5025)**

CAUSE The target drive number specified by the user in response to the prompt at initialization time does not correspond to a valid drive on the system.

ACTION Check the target drive number that was selected and make sure that there is a drive connected to the system that corresponds to that address. If so, ensure that the drive is powered up.

5026	`*** WARNING -- THE IDENTITY OF THE LINK HPFL CONTROLLER COULD NOT BE OBTAINED. SECTIONS 10 AND/OR 11 SHOULD BE RUN IN ORDER ISOLATE THE PROBLEM. MEANWHILE, ANY TESTS THAT REQUIRE A TARGET DRIVE WILL NOT BE PERFORMED. (HPFLWARN 5026)`
CAUSE	An attempt was made to identify the link drive controller via a well-known VC identify operation which failed. This operation is attempted at initialization time and the failure is only actually reported if the user selects LINK in response to the prompt asking for a drive to test.
ACTION	The directions given in the text of the message should be followed which basically directs the user to run the trouble-tree sections for more information.

5027	`*** WARNING -- THE IDENTITY OF THE DRIVES CONNECTED TO THE LINK HPFL CONTROLLER COULD NOT BE OBTAINED. SECTIONS 10 AND/OR 11 SHOULD BE RUN IN ORDER TO ISOLATE THE PROBLEM. (HPFLWARN 5027)`
CAUSE	An attempt was made to identify the drive controllers connected to the HPFL Interface card via a rollcall operation which failed. This operation is attempted at initialization time.
ACTION	The directions given in the text of the message should be followed which basically directs the user to run the trouble-tree sections for more information.

5028	`*** WARNING -- TESTING OF THE HP-FL CARD ONLY WAS SELECTED. AT LEAST ONE DRIVE MUST HAVE BEEN SELECTED FOR TESTING IN ORDER TO RUN THIS STEP. NO OPERATION WAS PERFORMED. (HPFLWARN 5028)`
CAUSE	A step was selected that performs an well-known vc operation. However, the user specifically requested that no drive tests be performed by selecting the NONE option to the initialization prompt asking which drives should be tested.
ACTION	Since the cause was simply selecting no drives to test, re-run the diagnostic and select a drive to test during initialization (via any option other than NONE).

5029	`*** SPECIFIED DRIVE NUMBER IS INVALID. A VALID DRIVE NUMBER MUST BE IN THE RANGE OF 0 TO 7. (HPFLERR 5029)`
CAUSE	The user input "drive n" in response to the prompt asking which drive to test and "n" was a valid number but not in the range of 0 to 7.
ACTION	Input a valid drive number in the range 0 to 7.

4

HP-IB Device Adapter Diagnostic

Introduction

The HP-IB Device Adapter Diagnostic (HPIBDIAG) is a diagnostic system program that provides the user or the HP-IB Device Adapter Manager (DAM) with the ability to test the functionality of the HP-IB Device Adapter. The diagnostic runs under MPE/iX or HP-UX on any HP Precision Architecture computer system from any system terminal. The diagnostic has no interactive commands, but the user can specify which sections and steps are to be run. The user can also set test parameters to control the handling of error messages and to select the number of test executions to be run. The HP-IB Device Adapter Diagnostic can also be invoked by the I/O subsystem during catastrophic errors for auto-diagnostic purposes (MPE/iX only). HP-IB Device Adapter functionality is restored by replacing the HP-IB Device Adapter PCA which is itself a Field Replaceable Unit (FRU).

Defects and Enhancements

Submit defect reports and enhancement requests for this diagnostic through the STARS database referencing product number 30600-10011.

Minimum Configuration

The hardware required to run the diagnostic consists of an HP Precision Architecture RISC computer system which is up and running on either the HP-UX or MPE/iX operating system.

Auto-Diagnostics

The HP-IB device adapter diagnostic program can be invoked (by the MPE/iX I/O system on catastrophic errors) for auto-diagnostic purposes. In auto-diagnostic mode, the HP-IB diagnostic program will execute the following sections and steps:

Section 3	Identify
Section 4	Loopback
Section 5	Selftest

Operating Instructions

There is no security level checking mechanism within HPIBDIAG. The DUI checks the user's security level before initiating HPIBDIAG. Refer to the Security section on DUI for a detailed description of user capabilities.

Default Tests

If the user does not specify the sections and steps to be run, the default sections will be executed:

Section 3	Identify
Section 4	Loopback
Section 5	Selftest

RUN Command

To bring up the Online Diagnostic subsystem, enter the following command to the system prompt:

```
sysdiag
```

The system responds with the following prompt indicating that access has been gained to the Online Diagnostic Subsystem:

```
DUI >
```

Note

The device to be tested must be powered up and on line. The physical device location (pdev) shown below matches the same device shown on the "typical A1002A" system configuration, described in the chapter on DUI. The pdev value entered must be correct for the system being tested.

For example, to run the diagnostic in an MPE/iX environment, you might enter:

```
DUI > RUN HPIBDIAG pdev=4.2.3 <RUN Command Options>
                           |             |
                           |       none required for
                           |       default test suite
                           |
             insert physical location of
             device adapter to be tested here;
             alternatively, for MPE/iX,
             type the ldev number;
             for HP-UX, type the devfile name
```

Typing HELP causes a summary of the DUI and its commands to be printed. Refer to the DUI Section for details.

Test Execution

The diagnostic displays a header and welcome message.

Following the header, HPIBDIAG will call a program service routine to test the I/O path between the SPU and the device adapter. This helps the user locate a critical failure or a corrupt data path between the host system and the device adapter. If the status returned from this procedure call is "fail", an error message will be output:

```
There is a problem in the path to the device adapter.
```

`HPIBDIAG` will continue if possible. The user should then troubleshoot all hardware between the device adapter and CPU/MEMORY. This would include all buses, bus converters, and channel adapters along with their power supplies.

If the path between the SPU and the device adapter is functional, the test sections and steps specified by the user will be executed and the results will be output. If the user has not specified any sections or steps to be run, the default sections will be run by **HPIBDIAG**. They are Sections 3, 4, and 5 (IDENTIFY, LOOPBACK and SELFTEST). These test sections are described in the "Test Section Descriptions".

If the ERRORCOUNT option of the **RUN** command is specified to a limit by the user and the number of errors generated by the HP-IB device adapter diagnostic reach that limit, the following message will be output:

```
More errors encountered than specified in the errcount.
```

The diagnostic will terminate execution immediately upon displaying the above message.

If the ERRONLY option of the RUN command is set, only error messages will be displayed to the user. If the ERRPAUSE option of the RUN command was set "on", then the diagnostic will stop after each error is generated and ask the user if the test should continue:

```
CONTINUE (YES/NO)?
```

If the response is "Y" then the test will be resumed (if possible), and if the response is "N", the diagnostic will terminate. If the section and steps specified by the user were executed the number of times specified in the **LOOP** option of the **RUN** command without the number of errors exceeding the **ERRNUM** value, the diagnostic will terminate normally and the following message will appear:

```
HPIBDIAG Exiting . . .
```

Control will then return to the Online Diagnostic System.

```
DUI >
```

Test Section Descriptions

HPIBDIAG consists of five diagnostic program sections:

```
Section 3          Identify
Section 4          Loopback
Section 5          Selftest
Section 6          Status
   Step 10            Request Status
   Step 11            Decode Status
Section 12            Rollcall
```

A description of each section and step will be given, along with the expected output from that section and step.

Section 3—IDENTIFY

This section of the diagnostic issues an IDY (Identify) command to the HP-IB device adapter. The response from hardware will be decoded into various pieces of information such as device adapter identification code, firmware identification, and firmware revision level. HPIBDIAG will report the firmware identification and hardware date code.

```
Section 3 - IDENTIFY

   The Identify was successful
   The device adapter; identifier number is nnn.
   The device adapter firmware ID is nnn.
   The device adapter date code is nnn.
   The device adapter hardware revision number is nnn.

End of Section 3 - IDENTIFY
```

Section 4—LOOPBACK

This section tests the data path to the HP-IB device adapter. The test is performed by sending patterns of 1's and 0's to the device adapter card's buffer and back again. The following is executed:

```
Section 4 - LOOPBACK
   External loopback of the DEVICE ADAPTER completed.
End of Section 4 - LOOPBACK
```

Section 5—SELFTEST

This section reports the results of the selftest as a GO/NO-GO status. If the HP-IB device adapter selftest fails then the device adapter itself must be replaced. The following message is displayed if the test is successful.

```
Section 5 - SELFTEST
   Selftest of HPIB DEVICE ADAPTER completed successfully.
End of Section 5 - SELFTEST
```

Section 6—STATUS

This diagnostic section obtains and decodes the status of the HP-IB device adapter hardware. Two steps are available:

Step 10 **Request Status:** `HPIBDIAG` will attempt to read the HP-IB device adapter card status, if successfully done, `HPIBDIAG` will return the value without decoding it.

Step 11 **Decode Status:** `HPIBDIAG` will decode the format of the HP-IB device adapter card status bits, determine the meaning of the hardware status, then return the messages according to the decoded results.

The following is executed:

```
Section 6 -- DEVICE STATUS

   Step 10 -- Read Status
      Device Adapter status has been read successfully.
   End of step 10 - Read Status

   Step 11 -- Read Status
      The current hardware status for the HPIB DEVICE ADAPTER is:
      -------------------------------------------------------------
      << status message >>
   End of Step 11 -- DECODE STATUS

End of Section 6 - DEVICE STATUS
```

Section 12—ROLLCALL

This section returns the information about the connection profile of the HP-IB device adapter being tested. The user is recommended to run all other diagnostics before running rollcall. If there is any malfunction of the HP-IB device adapter hardware, the information returned from this section may not be valid. The following is executed:

```
Section 12 - ROLLCALL
      Device Address
      0  1  2  3  4  5  6  7
      ----------------------
      << Rollcall array >>
```

A '1' under an address number means that a device was found at that address. A '0' means that no device was found at that address.

```
End of Section 12 - ROLLCALL
```

To decode the device array identifier use the SYSMAP utility.

Error Messages

The following are error messages which may be encountered during the execution of **HPIBDIAG**. System dependent error messages may also be displayed by the diagnostic system; error messages without the trailer (HDIAGERR #) are, in general, generated by the diagnostic system. However, please note that three **HPIBDIAG** informational messages (101, 402, and 403) will only appear when there has been an error and so are included below. These three messages do not have the error trailer. The "!" indicates that a parameter of some sort replaces the exclamation point when the message is displayed.

101 **There may be a problem in the path to the device adapter.**

CAUSE The call to the program service io_path_test was not successful.

ACTION io_path_test will have printed its own error stating the particular test which failed and the PDEV of the device which failed. That device should be tested further before continuing.

---

402 **ERROR IN TRANSMISSION DETECTED DURING READ LOOPBACK TEST:**

```
                                                        Bit Positions
                      Octal Value       Octal Value     In Error
            Byte #    Transmitted       Received          01234567
            ======    ===========       ========        =============
403           !           !                 !                 !
```

CAUSE Either writing to or reading back from the HP-IB failed.

ACTION The bit positions in error should give some indication of where the problem exits; run selftest and check the cabling.

5501 **UNABLE TO SELECT THE DEVICE (HDIAGERR 5501)**

CAUSE The diagnostic could not obtain access to the HP-IB; the reason will be stated in a preceding error message.

ACTION Refer to the action to be taken which is associated with the preceding error.

5502 **UNABLE TO GET INPUT BUFFER (HDIAGERR 5502)**

CAUSE A software error has occurred.

ACTION Please report the error to support personnel.

5503	**UNABLE TO GET OUTPUT BUFFER (HDIAGERR 5503)**
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5504	**UNABLE TO GET HARDWARE STATUS BUFFER (HDIAGERR 5504)**
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5505	**FAILED TO RETRIEVE THE HARDWARE REVISION NUMBER (HDIAGERR 5505)**
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5601	**UNABLE TO MAKE STRING FROM NUMBER (HDIAGERR 5601)**
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5602	**UNABLE TO PULL BITS FROM A 32 BIT INTEGER (HDIAGERR 5602)**
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

6001	**AN UNEXPECTED STATUS WAS RECEIVED (HDIAGERR 6001)**
CAUSE	A completely unanticipated failure occurred somewhere. The actual non-successful status which triggered this error message will be printed.
ACTION	Refer to the action associated with the message which will be printed immediately after this one.

6201	**ROLLCALL FUNCTION TO DEVICE ADAPTER FAILED (HDIAGERR 6201)**
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.

6301	**IDENTIFY FUNCTION TO DEVICE ADAPTER FAILED (HDIAGERR 6301)**
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.
6401	**LOOPBACK FUNCTION TO DEVICE ADAPTER FAILED (HDIAGERR 6401)**
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one. Please note that whatever the problem is it precluded writing data to the hpib, reading it back and comparing it. If the loopback 'fails' because the data read from it is not identical to the data written to it a different message will be given.
ACTION	Refer to the action associated with the succeeding message.
6501	**SELFTEST FUNCTION TO DEVICE ADAPTER FAILED (HDIAGERR 6501)**
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.
6601	**REQUEST STATUS FUNCTION TO DEVICE ADAPTER FAILED (HDIAGERR 6601)**
CAUSE	Unknown. The reason for the failure will be indicated by the message immediately succeeding this one.
ACTION	Refer to the action associated with the succeeding message.
6603	**UNABLE TO MOVE DATA FROM OUT_DATA_BUFFER (HDIAGERR 6603)**
CAUSE	A software error has occurred.
ACTION	Please report the error to support personnel.

5

HP-IB Device Adapter Diagnostic

Introduction

The HP-IB Device Adapter Diagnostic (`HPIBDAD`) is part of the online diagnostic package. It is designed to provide its user with a means of determining if the specified HP-IB Device Adapter (DA) and its related hardware are operating properly, and if not, which FRU should be replaced. There are a variety of tests which the user can run to determine the source of a problem. Some of the tests require writing and reading data to and from (respectively) an external device (e.g., a disk drive). Tests that require such action need only be run during exceptional situations and are included so that the diagnostic can help in determining the source of a problem. However, these tests may not be appropriate to run in all environments. For example, if the HP-IB DA to be tested is connected to a boot device, executing the write/read tests on that device could write over the operating system code. The result of such an action could be a system crash.

Note

Any interaction with a device on the HP-IB is administered by the diagnostic user, NOT the diagnostic. Therefore, extreme caution should be taken by the user before such tasks are undertaken.

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10028.

Minimum Configuration

The hardware required to run HPIBDAD consists of a Hewlett-Packard Precision Bus (PB) HP-IB Device Adapter (part number 28650A), as well as an HP 815 computer system with all the necessary equipment to bring the operating system (HP-UX) up and running.

However, in order to fully test the HP-IB Device Adapter, an HP-IB Talker/Listener/Controller device is required to be connected to the HP-IB. More specifically, there must be a device connected to the HP-IB which has the ability to Talk, and whose output data can be verified (e.g., a digital multimeter which outputs a known voltage); there must be a device connected to the HP-IB which has the ability to Listen, and whose input data (command) can be verified (e.g., a plotter which is given the "command" to lift its pen); and there must be a device that can take control of the HP-IB. It doesn't matter whether or not all of this functionality is implemented in a single device or in more than one device.

In order to run HPIBDAD, the online diagnostic system must be present, along with the portability interface routines. Also, the diagnostic must guarantee that regardless of what condition the HP-IB Device Adapter (DA) to be tested is in, the system will not "crash" by running the diagnostic on it.

Other software that must be present in the system to execute HPIBDAD is the following: 1) the HP-IB Device Access Routines (DAR) which act as an interface between the diagnostic and the Logical Device Manager (LDM), 2) the LDM which is the interface between the DAR and low-level driver, and 3) the HP-IB Device Adapter Manager (DAM) which is the low-level driver to the HP-IB DA. Note that the DAR is technically part of the portability interface, as far as online module structure is concerned. Also note that in this document, references to the DAM imply the LDM as well, unless otherwise noted.

Operating Instructions

HPIBDAD is accessed by the user via the Diagnostic User Interface (DUI).

Default Tests

The default sections and steps for this diagnostic are:

```
Section 3   Identify

Section 6   Status

  Step 61     Preliminary Internal State Diagnosis

  Step 62     Read HP-IB Interface Chip STATUS Register

  Step 63     Read HP-IB Interface Chip CONTROL Register

  Step 64     Read HP-IB Interface Chip ADDRESS Register

  Step 65     Read HP-IB Interface Chip PP/ID_BYTE Registers

  Step 66     Read HPIB_STATUS Register

  Step 67     Read BUS_STATUS Register
```

RUN Command

To bring up the Online Diagnostic subsystem, enter the following command to the HP-UX system prompt:

```
% sysdiag
```

The diagnostic subsystem responds with the following prompt indicating that access has been granted to the user:

```
DUI >
```

Typing HELP causes a summary of the DUI function and its commands to appear on the screen.

Note The device to be tested must be powered up and on line. The physical device location (pdev) shown below matches the same device shown on the "typical A1002A" system configuration, described in the chapter on DUI. The pdev value entered must be correct for the system being tested.

After the online diagnostics system has been started, this diagnostic can be executed using the command:

```
RUN HPIBDAD
```

All online diagnostics RUN string parameters are acceptable when executing this diagnostic. All parameters available in the run command are acceptable as parameters when executing this diagnostic.

Note

All of the sections in this diagnostic can be executed from any terminal, even if a specific test requires the user to have the capability to run destructive tests. This implementation therefore allows the diagnostic to be run from a remote terminal; however, there may be system limitations that would not allow the use of a remote terminal.

Test Execution

When HPIBDAD is invoked, a header and welcome message will be displayed. After the header and welcome message are displayed via a program services call, the diagnostic will call another program services routine in order to obtain access to the device that was selected for testing (in addition to setting up the sections and steps to be run).

This routine will exit with its *status* parameter (passed by reference) being any one of three possible values. The first of which is *successful*. This indicates that all sections and steps have been validated and that the system granted access to the device.

The second possible value is *dssd_device_in_use*. If this value is returned, it indicates that the system did not grant access to the device. If this happens, the following error message will be issued by the diagnostic:

```
*** ERROR -- HP-IB DEVICE ADAPTER ALREADY IN USE BY THE
***          DIAGNOSTIC SYSTEM.                          (HDADERR 5000)
***
***          Someone has already gained exclusive rights to the
***          HP-IB Device Adapter that you requested, and it is illegal
***          to have two copies of the HPIBDAD diagnosing the same HP-IB
***          Device Adapter simultaneously.
```

The diagnostic will terminate execution after outputting this error message.

The third possible status value is *dssd_internal_error*. When this value is active upon exiting the subroutine, it indicates that an error such as no device adapter at the specified LDEV was found. The online diagnostics themselves will output the error message for this situation, *not* HPIBDAD. The diagnostic will terminate upon regaining control.

If all went well up to this point, the sections and steps specified by the user will be executed and the results displayed. If the user did not specify any sections/steps to be run, the default sections and steps will be executed (Sections 3 and 6). If at any time, the number of errors generated exceeds the limit specified by the user in the *errcount* parameter (of the DUI *run* command), the following message will be output:

```
*** WARNING -- The maximum specified number of error occurrences has
***            been exceeded.                           (HDADWARN 6000)
```

The diagnostic will then terminate its execution. If the *errpause* parameter of the **run** command was assigned to "on", then the diagnostic will stop after each error is generated and ask the user if the testing should continue. The prompt that will be displayed is as follows:

```
Do you wish to continue? (Y/N) [Y] :
```

If the response is "Y", then the testing will resume (if possible), and if the response is "N", the diagnostic will terminate its execution. If the sections and steps specified by the user were executed the number of times specified in the *loop* parameter of the **run** command without the number of errors exceeding the *errcount* value, the diagnostic will terminate normally.

At any time that the diagnostic is prompting the user for information, the user may enter "exit" to terminate its execution, or enter "suspend" to temporarily suspend its execution. Either the entire word or any number of characters which uniquely identify the respective language localized command may be entered. Moreover, the letters entered may be in any combination of upper and lower case characters. If the user exits in this fashion, the following message is displayed:

```
... Exiting HPIBDAD per your request.
```

If the user temporarily suspends execution in this manner, the message that will be displayed is as follows:

```
... HPIBDAD suspended per your request.
```

The user can then perform tasks through the Diagnostic User Interface (DUI) and subsequently resume execution of HPIBDAD, or he/she can abort the HPIBDAD entirely.

Note

In situations such as timeouts, this diagnostic will inform the user which operation was taking place when the timeout occurred; moreover, the user will be given a list of commands which executed successfully (if any). It is believed that the value of this information outweighs the risk of inundating the user with output.

Detailed Test Descriptions

The remainder of this section discusses each section and step in detail. As a quick reference, the following table was included to list all of the sections and steps available for use in HPIBDAD.

```
  Section No.        Diagnostic Function
=============================================================================

       1             More Help

       2             Reset

       3             Identify

       4             Local Loopback
                        Step 42 - Loopback from PB Interface Chip
                        Step 43 - Loopback from HP-IB Interface Chip

       5             Hardware Test

       6             Status
                        Step 61 - Preliminary Internal State Diagnosis
                        Step 62 - Read HP-IB Interface Chip STATUS Register
                        Step 63 - Read HP-IB Interface Chip CONTROL Register
                        Step 64 - Read HP-IB Interface Chip ADDRESS Register
                        Step 65 - Read HP-IB Interface Chip
                                     PP/ID_BYTE Registers
                        Step 66 - Read HPIB_STATUS Register
                        Step 67 - Read BUS_STATUS Register

      10             Register Level Input/Output Transactions

      11             Data/Command Transaction on HP-IB
```

Section 1—More Help

```
Minimum Mode Required        : Normal

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

`More Help` is an interactive section which allows the user to obtain more information about a particular section than is given when typing **help hpibdad** at the DUI prompt. This is needed because it is not desirable to spew large help screens at the user when he/she is looking for general help, but it is desirable to give more information about certain sections when requested.

This section allows all users from any terminal to obtain the additional information that they request.

Possible Output Messages:

```
Section 1 -- More Help

    This Section allows you to get more information on any of the Sections
    [1..6, 10, 11] of this diagnostic.  Please indicate the number of the
    section for which you require more information.  Entering a lone <CR> to
    the prompt exits this section.

    More Help (1..6, 10, 11, <CR>) : [Return]

End of Section 1 -- More Help
```

If the user enters a section number in response to the prompt, the pertinent information would be displayed for the user to read. The numbering scheme used for the help messages is as follows: the number of the message is equal to *(section_number) * 100 + 10000*. This allows for 100 messages per section. Note that when multiple messages exist for a given section, all of the corresponding messages are displayed.

Note Informational messages are indented four spaces from the left margin and implement standard capitalization rules. This is done in order to make the error/warning messages more obvious to the user (which are not indented from the left margin, and are preceded by three asterisks).

Possible Error/Warning Messages:

If the user inputs a number of a section that is not implemented (i.e. not in the set {1,2,3,4,5,6,10,11}), then the following message is output:

```
*** WARNING -- Invalid response.  Please answer the question with one
***            of the choices given.                          (HDADWARN 6001)
```

```
More Help (1..6, 10, 11, <CR>) :
```

Note that the user is prompted again for input.

HP-IB DAR Operations Used:

```
None specified by the diagnostic.
```

Section 2—Reset

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

This section informs the Device Adapter Manager (DAM)—via the Device Access Routines—to reset the HP-IB Device Adapter (DA) and DAM to its power-on state. The DAM will then configure the card with information it maintains internally.

Possible Output Messages:

```
Section 2 -- Reset

   NO ERRORS DETECTED while resetting the device adapter.

End of Section 2 -- Reset
```

Possible Error/Warning Messages:

None specified by the diagnostic.

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_RESET
HPIB_UNLOCK
```

Section 3—Identify

```
Minimum Mode Required         : Destructive

Terminal Used for Execution : Any

In Default Set?               : Yes

In Auto-diagnostic Set?       : No
```

Identify issues an HPIB_IDENTIFY command to the HP-IB DAR, which acquires the requested information from the HP-IB DAM, which in turn acquires most of the information from the HP-IB DA's IODC (the only information not retrieved from the IODC is the DAM, LDM, and DAR version codes). The diagnostic then decodes the information obtained and displays it in a manner that is informative to the user. This section can be used to determine the HP-IB DA's hardware and software versions, as well as the version of the DAM, LDM, and DAR being used. Moreover, a checksum is calculated by the system on the appropriate IODC information when this section is executed. If the checksum test fails, an error message will be displayed. This section has an added benefit in that if it executes successfully, the path from the diagnostic to the HP-IB DA is known to be at least partially functional.

Possible Output Messages:

```
Section 3 -- Identify


   Hardware Version                   : 0x?

   Soft Physical Address Capability : 0x80

   Type of Module                     : 4 (Type A DMA I/O Adapter)

   Software Version                   : 0x?

   IODC Revision                      : ?

   ... Checksum Verified.

   Device Adapter Manager Version    : ?

   Logical Device Manager Version    : ?

   Device Access Routine Version     : ?

   Note: The "0x" prefix is used to specify that the respective number
         is in hexadecimal format.  Also note that version numbers
         depicted by '---' implies that the actual version number was not
         accessible.


End of Section 3 -- Identify
```

Note The Hardware Version, Software Version, IODC Revision, Device Adapter Manager Version, Logical Device Manager Version, and Device Access Routine Version fields may vary in time, therefore cannot be explicitly specified within this document.

Possible Error/Warning Messages:

If the Soft Physical Address Capability is not 0, the following message is displayed:

```
*** WARNING -- Soft Physical Address Capability = 0x!,
***            I expected  0x80.                          (HDADWARN 6010)
```

If the module type value returned is not that of a Type A DMA I/O Adapter, the following message is displayed:

```
*** WARNING -- Type of Module = ! (UNKNOWN PRODUCT),
***            I expected a 4 (TP_A_DMA).                 (HDADWARN 6011)
```

If the checksum does not equal zero, the following error message is displayed:

```
 *** ERROR -- IODC CHECKSUM FAILED.                       (HDADERR 5005)
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_IDENTIFY
HPIB_UNLOCK
```

Section 4—Local Loopback

```
Minimum Mode Required        : Destructive

Terminal Used for Execution  : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

The local loopback tests will determine the operational status of the backplane and midplane of the HP-IB Device Adapter. This will be accomplished by writing and reading data to and from (respectively) the HP-IB DA via HPIBDAD—in an "onion skin" fashion. For this reason, the local loopback tests are divided into two steps according to how "deeply" they interact with the HP-IB DA.

If this section is run without the desired steps being explicitly specified, Steps 42 and 43 will both be run by default.

Note

Since the HP-IB requires bidirectional drivers on the frontplane, the frontplane cannot be tested via local loopback. The only way to test the frontplane would be with either an active loopback hood, or by transmitting data to and from an external device. Since the former option is not very practical, the latter option was chosen for this diagnostic. However, this functionality is NOT controlled by **HPIBDAD**, but rather is controlled by the user. For more information, please see Section 11, "Data/Command Transaction on HP-IB".

Step 42—Loopback from PB Interface Chip

In this step, data will be written to the PB Interface Chip and read back, testing the hardware layer "just below" the backplane (recall that the "onion skin" approach to diagnostics is implemented).

Note The chip is used to interface the PB backplane to the midplane of the HP-IB DA.

HPIBDAD will compare the data read with the data written, and display the appropriate messages.

Possible Output Messages:

```
Section 4 -- Local Loopback

   Step 42 - Loopback from PB Interface Chip

      NO ERRORS DETECTED while executing loopback from the PB
      Interface Chip.

   End of Step 42 - Loopback from PB Interface Chip

End of Section 4 -- Local Loopback
```

Possible Error/Warning Messages:

```
*** ERROR -- PB INTERFACE CHIP (PBIC) LOOPBACK FAILED.          (HDADERR 5010)
***
***         Data written to PBIC (in hex)        Data read from PBIC (in hex)
***         -----------------------------        ----------------------------
***                       !                                     !
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_WRITE_REG
HPIB_READ_REG
HPIB_RESET
HPIB_UNLOCK
```

Step 43—Loopback from HP-IB Interface Chip

In this step, data will be written to the HP-IB Interface Chip and read back, testing the hardware up to but not including the frontplane transceivers.

Note The chip is an HP-IB Talker/Listener/Controller.

HPIBDAD will compare the data read with the data written, and display the appropriate messages.

Possible Output Messages:

```
Section 4 -- Local Loopback

   Step 43 - Loopback from HP-IB Interface Chip

      NO ERRORS DETECTED while executing loopback from the HP-IB
      Interface Chip.

   End of Step 43 - Loopback from HP-IB Interface Chip

End of Section 4 -- Local Loopback
```

Possible Error/Warning Messages:

```
*** ERROR -- HP-IB INTERFACE CHIP (HIC) LOOPBACK FAILED.       (HDADERR 5020)
***
***         Data written to HIC (in hex)        Data read from HIC (in hex)
***         ----------------------------        ---------------------------
***                     !                                   !
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_WRITE_REG
HPIB_READ_REG
HPIB_RESET
HPIB_UNLOCK
```

Section 5—Hardware Test

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

Executing this section will perform a thorough hardware test on the HP-IB DA. Not only will the PB Interface and the frontplane interface chips be tested, but the glue logic and backplane transceivers will also be tested. The only components that will be untested after this section is executed is the glue logic that is inaccessible to HPIBDAD (obviously), and the frontplane transceivers. The reason why the frontplane transceivers will remain untested is because there is no straightforward way to test them. The only way they can be tested is by going off the card, and since this functionality would be impractical to include in HPIBDAD, it was chosen not to test the frontplane transceivers. However, the user does have the capability to do I/O with devices connected to the HP-IB; therefore, it is not impossible to test the frontplane transceivers with this diagnostic software.

The user will be given the capability to go onto the HP-IB since he/she should know what devices are "out there" and should also know their respective command sets; whereas to include the capability for HPIBDAD to be able to identify a particular device (if possible) and be able to communicate with it would be a supererogatory effort on the part of the diagnostic writer.

Since this hardware testing brings the card off-line, it should only be done when absolutely necessary.

Possible Output Messages:

```
Section 5 -- Hardware Test

   NO ERRORS DETECTED while testing the PB Interface Chip's
        IO_EIM register.

   NO ERRORS DETECTED while testing the PB Interface Chip's
        IO_DMA_LINK register.

   NO ERRORS DETECTED while testing the PB Interface Chip's
        IO_DMA_COUNT register.

   NO ERRORS DETECTED while testing the HP-IB Interface Chip's
        INTERRUPTING_MASK register.

   NO ERRORS DETECTED while testing the HP-IB Interface Chip's
        HP-IB Interface Chip CONTROL register.

   NO ERRORS DETECTED while testing the HP-IB Interface Chip's
        ADDRESS register.

   NO ERRORS DETECTED while testing the HP-IB Interface Chip's
        PARALLEL_POLL_MASK/FIRST_ID_BYTE register.
   NO ERRORS DETECTED while testing the HP-IB Interface Chip's
        PARALLEL_POLL_SENSE/SECOND_ID_BYTE register.

   NO ERRORS DETECTED during the "HPIB_STATUS Register" Test
   NO ERRORS DETECTED during the "IFC" Test
   NO ERRORS DETECTED during the "Talk/Listen" Test
   NO ERRORS DETECTED during the "REN" Test
   NO ERRORS DETECTED during the "FIFO" Test
   NO ERRORS DETECTED during the "SRQ" Test
   NO ERRORS DETECTED during the "Parallel Poll" Test
   NO ERRORS DETECTED during the "Secondary Address" Test
   NO ERRORS DETECTED during the "CRC" Test
   NO ERRORS DETECTED "DMA Test Number 1"
   NO ERRORS DETECTED "DMA Test Number 2"
   NO ERRORS DETECTED "DMA Test Number 3"
   NO ERRORS DETECTED "DMA Test Number 4"
   NO ERRORS DETECTED "DMA Test Number 5"
   NO ERRORS DETECTED "DMA Test Number 6"
   NO ERRORS DETECTED "DMA Test Number 7"
   NO ERRORS DETECTED "DMA Test Number 8"
   NO ERRORS DETECTED the "GET/HP-IB Interface Chip Interrupt Test"
   NO ERRORS DETECTED the "On-Line" Test
   NO ERRORS DETECTED the "IFC Interrupt" Test

End of Section 5 -- Hardware Test
```

Possible Error/Warning Messages:

```
*** WARNING -- The device adapter under test is NOT the System
***            Controller, therefore the "On-Line Test" cannot be
***            executed.                                      (HDADWARN 6015)

*** WARNING -- The device adapter under test is NOT the System
***            Controller, therefore the "IFC Interrupt Test" cannot
***            be executed.                                   (HDADWARN 6016)

*** ERROR -- PB INTERFACE CHIP (PBIC) TEST FAILED.            (HDADERR 5030)
***          ! register failed the "Register Verification" test.
***
***          Data written to PBIC (in hex)      Data read from PBIC (in hex)
***          -----------------------------      ----------------------------

*** ERROR -- HP-IB INTERFACE CHIP (HIC) TEST FAILED.          (HDADERR 5031)
***          ! register failed the
***          "Register Verification" test.
***
***          Data written to HIC (in hex)      Data read from HIC (in hex)
***          ----------------------------      ---------------------------

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5035)
***          The interface chip "IFC" test failed.

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5036)
***          The interface chip "Talk/Listen" test failed.

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5037)
***          The interface chip "REN" test failed.

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5038)
***          The interface chip "FIFO" test failed.

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5039)
***          The interface chip "SRQ" test failed.

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5040)
***          The interface chip "Parallel Poll" test failed.

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5041)
***          The interface chip "Secondary Address" test failed.

*** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5042)
***          The interface chip "CRC" test failed.

*** ERROR -- HPIB_CONTROL/STATUS REGISTER SET VERIFICATION
***          TEST FAILED.                                     (HDADERR 5045)

*** ERROR -- GROUP EXECUTE TRIGGER OR HP-IB INTERFACE CHIP'S INTERRUPT
```

```
***         CIRCUITRY FAILED TEST.                              (HDADERR 5046)

*** ERROR -- ON-LINE TEST FAILED.                               (HDADERR 5047)
***
***         This test is targeted to check the frontplane's General Interface
***         Management Lines (control transceiver), and BUS_STATUS register.

*** ERROR -- IFC INTERRUPT TEST FAILED.                         (HDADERR 5048)
***
***         This test is targeted to check the frontplane's IFC
***         interrupt circuitry.

*** ERROR -- DMA TEST NUMBER ! FAILED.                          (HDADERR 5049)
***         Data was corrupted.

*** ERROR -- DMA TEST NUMBER ! FAILED.                          (HDADERR 5050)
***         DMA termination condition incorrect.

*** ERROR -- DMA TEST NUMBER ! FAILED.                          (HDADERR 5051)
***         Incorrect residue value in the IO_DMA_COUNT register.

*** ERROR -- DMA TEST NUMBER ! FAILED.                          (HDADERR 5052)
***         Internal Software Error.

*** ERROR -- DMA TEST NUMBER ! FAILED.                          (HDADERR 5053)
***         Inconsistency within PB Interface Chip with respect to
***         reporting "DMA length conflict" status.
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_WRITE_REG
HPIB_READ_REG
HPIB_OUTPUT
HPIB_INPUT
HPIB_RESET
HPIB_UNLOCK
```

Section 6—Status

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : Yes

In Auto-diagnostic Set?      : No
```

This section is broken down into seven steps so that the diagnostician can convey to HPIBDAD precisely what status information he/she wishes to examine. Moreover, the diagnostician can have a very thorough status report of the HP-IB Device Adapter if he/she desires it by executing all seven steps. In order to make this ample supply of status information easier for the diagnostician to assimilate, it is fully decoded and transformed into user-friendly messages by HPIBDAD before being displayed to the diagnostic user.

If this section is run without the desired steps being explicitly specified, Steps 61, 66, and 67 will run by default.

Step 61—Preliminary Internal Status Diagnosis

Executing this step instructs the diagnostic to gather and interpret the necessary data in order to report a preliminary device adapter internal state diagnosis. The information gathering portion of this step consists of Device Access Routine (DAR) calls—register reads—which return the contents of the following registers:

- PB Interface Chip IO_STATUS Register
- PB Interface Chip DIAGNOSTIC_STATUS Register
- HP-IB Interface Chip INTERRUPTING_CONDITIONS Register
- HP-IB Interface Chip INTERRUPT_MASK Register

The interpretation phase of this data is considerably more complex and consists primarily of the following functions:

- Decode and report (in a user-friendly fashion) the information maintained by the various registers.
- If more than one register reports the same status condition (e.g. an interrupt is requested), `HPIBDAD` will compare the respective bits of those registers to check status consistency within the device adapter. Any inconsistency will be reported to the user as an error condition.
- If an interrupt is requested by the frontplane interface chip, the diagnostic will determine the interrupt condition and report it. However, if `HPIBDAD` discovers that the particular interrupt should have been masked because the INTERRUPT_MASK register contains a "0" at the respective bit position, an error message will be issued.
- If an interrupt is not requested, but the contents of INTERRUPTING_CONDITIONS and INTERRUPT_MASK indicate an interrupt request should have been issued, an error message will be displayed to the user.

Possible Output Messages:

```
Section 6 -- Status

   Step 61 - Preliminary Internal Status Diagnosis

      PB Interface Chip IO_STATUS Register Data:

         An interrupt has NOT occurred since the last write to
         IO_COMMAND.                                         (Bit  0 = 0)

         An interrupt message HAS been sent since the last write
         to the IO_COMMAND register.                         (Bit  0 = 1)

         An interrupt message was NOT issued due to circuitry external
         to the PB Interface Chip.                           (Bit  2 = 0)

         An interrupt HAS been issued due to circuitry external to
         the PB Interface Chip.                              (Bit  2 = 1)


         Transfer NOT completed.                             (Bit  3 = 0)

         Transfer completed.                                 (Bit  3 = 1)

         A soft error (length conflict) has NOT occurred since the
         last write to the IO_COMMAND register.              (Bit 22 = 0)

         A fatal error has NOT occurred since the last write to
         the IO_COMMAND register.                            (Bit 24 = 0)

         The device adapter is NOT ready for a new command.  (Bit 25 = 0)

         The device adapter IS ready for a new command.      (Bit 25 = 1)

         An interrupt message has NOT been sent since last
         ii_clear.                                           (Bit 26 = 0)

         An interrupt message HAS been sent since last
         ii_clear.                                           (Bit 26 = 1)

         The EOC bit of the IO_DMA_LINK register
         is NOT set.                                         (Bit 31 = 0)

         The EOC bit of the IO_DMA_LINK register IS set.     (Bit 31 = 1)


      PB Interface Chip DIAGNOSTIC_STATUS Register Data:

         Circuitry external to the PB Interface Chip HAS issued
           an interrupt request.                             (Bit  0 = 0)
```

```
   Circuitry external to the PB Interface Chip has NOT
     issued an interrupt request.                       (Bit  0 = 1)

   The PB Interface Chip did NOT generate a PB error since the
     last reset command.                                (Bit  1 = 1)

   DMA is DISABLED.                                     (Bit  4 = 0)
   DMA is ENABLED.                                      (Bit  4 = 1)

   Interrupt message transmission is DISABLED.          (Bit 19 = 0)
   Interrupt message transmission is ENABLED.           (Bit 19 = 1)

   The FIFO is NOT FULL.                                (Bit 20 = 0)
   The FIFO is FULL.                                    (Bit 20 = 1)

   The FIFO is NOT EMPTY.                               (Bit 21 = 0)
   The FIFO is EMPTY.                                   (Bit 21 = 1)

   There is 1 byte in the FIFO.                         (Bits 23..27)
   There are ! bytes in the FIFO.                       (Bits 23..27)


HP-IB Interface Chip INTERRUPTING_CONDITIONS Register Data:

   An interrupt is NOT pending.                         (Bit  0 = 0)
   An interrupt IS pending.                             (Bit  Q = 1)

   "Parity Error" interrupt recorded.                   (Bit  1 = 1)

   "Status Change" interrupt recorded.                  (Bit  8 = 1)

   "Processor Handshake Abort" interrupt recorded.      (Bit  9 = 1)

   "Parallel Poll Response" interrupt recorded.         (Bit 10 = 1)

   "Service Request" interrupt recorded.                (Bit 11 = 1)

   "FIFO Room Available" interrupt recorded.            (Bit 12 = 1)

   "FIFO Byte Available" interrupt recorded.            (Bit 13 = 1)

   "FIFO Idle" interrupt recorded.                      (Bit 14 = 1)

   "Device Clear" interrupt recorded.                   (Bit 15 = 1)


HP-IB Interface Chip INTERRUPT_MASK Register data:

   Interrupts are DISABLED.                             (Bit  0 = 0)
   Interrupts are ENABLED.                              (Bit  0 = 1)
```

```
"Parity Error" interrupt MASKED.                       (Bit  1 = 0)
"Parity Error" interrupt UNMASKED.                     (Bit  1 = 1)

"Status Change" interrupt MASKED.                      (Bit  8 = 0)
"Status Change" interrupt UNMASKED.                    (Bit  8 = 1)

"Processor Handshake Abort" interrupt MASKED.          (Bit  9 = 0)
"Processor Handshake Abort" interrupt UNMASKED.        (Bit  9 = 1)

"Parallel Poll Response" interrupt MASKED.             (Bit 10 = 0)
"Parallel Poll Response" interrupt UNMASKED.           (Bit 10 = 1)

"Service Request" interrupt MASKED.                    (Bit 11 = 0)
"Service Request" interrupt UNMASKED.                  (Bit 11 = 1)

"FIFO Room Available" interrupt MASKED.                (Bit 12 = 0)
"FIFO Room Available" interrupt UNMASKED.              (Bit 12 = 1)

"FIFO Byte Available" interrupt MASKED.                (Bit 13 = 0)
"FIFO Byte Available" interrupt UNMASKED.              (Bit 13 = 1)

"FIFO Idle" interrupt MASKED.                          (Bit 14 = 0)
"FIFO Idle" interrupt UNMASKED.                        (Bit 14 = 1)

"Device Clear" interrupt MASKED.                       (Bit 15 = 0)
"Device Clear" interrupt UNMASKED.                     (Bit 15 = 1)
```

```
A binary representation of the respective registers follows:
NOTE: Dashes represent undefined bits.

   ---------------------------------------------------------
   IO_STATUS register's image:

   Bit #:     0     4     8     12    16    20    24    28

   Value:     #### #### #### #### #### #### #### ####
   ---------------------------------------------------------
   DIAGNOSTIC_STATUS register's image:

   Bit #:     0     4     8     12    16    20    24    28

   Value:     #### #### #### #### #### #### #### ####
   ---------------------------------------------------------
   INTERRUPTING_CONDITIONS register's image:

   Bit #:     0     4     8     12    16    20    24    28

   Value:     #### #### #### #### #### #### #### ####
   ---------------------------------------------------------
```

```
            INTERRUPT_MASK register's image:

            Bit #:     0     4     8    12    16    20    24    28

            Value:     ####  ####  ####  ####  ####  ####  ####  ####
            ---------------------------------------------------------

      End of Step 61 - Preliminary Internal State Diagnosis

   End of Section 6 -- Status
```

Possible Error/Warning Messages:

```
*** WARNING -- The IO_STATUS register has one or more undefined bits
***            reading in as a 1 (I expected 0's).              (HDADWARN 6020)

*** WARNING -- A SOFT ERROR (length conflict) occurred since the last
***            write to the IO_COMMAND register.
***            (IO_STATUS Bit 22 = 1)                           (HDADWARN 6022)

*** WARNING -- A FATAL ERROR occurred since the last write to the
***            IO_COMMAND register.
***            (IO_STATUS Bit 24 = 1)                           (HDADWARN 6023)

*** WARNING -- PB Interface Chip HAS generated a PB bus error since
***            the last reset command.
***            (DIAGNOSTIC_STATUS Bit 1 = 0)                    (HDADWARN 6025)


*** ERROR -- PB INTERFACE CHIP FAILURE.                         (HDADERR 5060)
***
***          PB Interface Chip reported to have ! bytes in FIFO,
***          however only 24 bytes are available.

*** ERROR -- PB INTERFACE CHIP FAILURE.                         (HDADERR 5061)
***
***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) is set
***          (implying FIFO CNT should = 24); however, the FIFO CNT field
***          (Bits 23..27) = ! (decimal).

*** ERROR -- PB INTERFACE CHIP FAILURE.                         (HDADERR 5062)
***
***          DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) is set
***          (implying FIFO CNT should = 0); however, the FIFO CNT field
***          (Bits 23..27) = ! (decimal).

*** ERROR -- PB INTERFACE CHIP FAILURE.                         (HDADERR 5063)
***
***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) and
***          FIFO EMPTY bit (Bit 21) are both set.
```

```
*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                     (HDADERR 5064)
***
***          THE "INTERRUPT PENDING" BIT WAS INADVERTENTLY SET:
***            The INTERRUPT PENDING bit (Bit 0) of
***            INTERRUPTING_CONDITIONS = 1; however, interrupts are
***            disabled, depicted by the INTERRUPT ENABLE bit (Bit 0) of
***            INTERRUPT_MASK = 0.

*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                     (HDADERR 5065)
***
***          AN INTERRUPTING CONDITION BIT WAS INADVERTENTLY SET:
***           One or more bits of INTERRUPTING_CONDITIONS = 1, even though the
***           interrupts have been masked, depicted by the respective bits of
***           INTERRUPT_MASK = 0.

*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                     (HDADERR 5066)
***
***          THE "INTERRUPT PENDING" BIT FAILED TO BE SET:
***            One or more bits of INTERRUPTING_CONDITIONS and Bit 0 of
***            INTERRUPTING_MASK = 1; however, the INTERRUPT PENDING bit
***            (Bit 0) of INTERRUPTING_CONDITIONS = 0.

*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                     (HDADERR 5067)
***
***          THE "INTERRUPT PENDING" BIT WAS INADVERTENTLY SET:
***            No interrupting condition bit is set, however the
***            INTERRUPT PENDING bit (Bit 0) of INTERRUPT_CONDITIONS = 1.

*** ERROR -- INCONSISTENCY IN INTERRUPT STATUS.                (HDADERR 5068)
***
***          The "INTERRUPT PENDING" bit (Bit 0) of
***          INTERRUPTING_CONDITIONS = 1;
***          however, the interrupt input bit of the DIAGNOSTIC_STATUS
***          register (Bit 0) = 1, which indicates that the PB Interface
***          Chip did not recognize the interrupt request.

*** ERROR -- PB INTERFACE CHIP FAILURE.                        (HDADERR 5069)
***
***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) = 0
***          (implying FIFO CNT should be less than 24); however, the
***          FIFO CNT field (Bits 23..27) = ! (decimal).

*** ERROR -- PB INTERFACE CHIP FAILURE.                        (HDADERR 5070)
***
***          DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) = 0
***          (implying FIFO CNT should be greater than 0); however, the
***          FIFO CNT field (Bits 23..27) = ! (decimal).
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_READ_REG
HPIB_UNLOCK
```

Step 62—Read HP-IB Interface Chip STATUS Register

Executing this step will return to the user various status conditions of the HP-IB Interface Chip. Once the HP-IB Interface Chip's STATUS register contents have been decoded by HPIBDAD, user-friendly output illustrating the current status information will be displayed.

Note

CONTROL and ADDRESS registers will need to be accessed in order to fully decode the STATUS register. This is because the on-line/off-line status (within the ADDRESS register) is needed to decode the HP-IB SYSTEM CONTROLLER bit (Bit 12). Also, the TALK ALWAYS and LISTEN ALWAYS bits within the ADDRESS register are needed for error detection purposes. The status of the IFC VALUE bit within the CONTROL register is needed for error detection concerning the HP-IB CONTROLLER bit (Bit 11).

In addition to the frontplane interface chip's own registers required to fully decode this register, the Device Adapter STATUS register is required in order to detect faults within the the frontplane interface chip. Specifically, the SYSTEM CNTL bit of the Device Adapter STATUS register is required to determine whether or not the HP-IB SYSTEM CONTROLLER bit of this register should be set.

Possible Output Messages:

```
Section 6 -- Status

   Step 62 - Read HP-IB Interface Chip STATUS Register

      High order access field:                          (Bit  8 = !)
                                                        (Bit  9 = !)

      Device Adapter is NOT in the remote state.        (Bit 10 = 0)
      Device Adapter IS in the remote state.            (Bit 10 = 1)

      Device Adapter is NOT the current HP-IB Controller.   (Bit 11 = 0)
      Device Adapter IS the current HP-IB Controller.   (Bit 11 = 1)

      Device Adapter is NOT the HP-IB System Controller.    (Bit 12 = 0)
      Device Adapter IS the HP-IB System Controller.    (Bit 12 = 1)
      Device Adapter IS the HP-IB System Controller, however
        it is also OFF-LINE.                            (Bit 12 = 1)

      Device Adapter has NOT been addressed to Talk OR
        to Identify.                                    (Bit 13 = 0)
      Device Adapter HAS been addressed to Talk OR Identify. (Bit 13 = 1)

      Device Adapter has NOT been addressed to Listen.  (Bit 14 = 0)
      Device Adapter HAS been addressed to Listen.      (Bit 14 = 1)

      Device Adapter's outbound data is NOT frozen.     (Bit 15 = 0)
      Device Adapter's outbound data IS frozen.         (Bit 15 = 1)

      A binary representation of the respective register follows:
      NOTE: Dashes represent undefined bits.

         ---------------------------------------------------------
         HP-IB Interface Chip STATUS register's image:

         Bit #:     0    4    8    12   16   20   24   28

         Value:    #### #### #### #### #### #### #### ####
         ---------------------------------------------------------

   End of Step 62 - Read HP-IB Interface Chip STATUS Register

End of Section 6 -- Status
```

Possible Error/Warning Messages:

```
*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                        (HDADERR 5080)
***
***         HP-IB SYSTEM CONTROLLER bit (Bit 12) should be set because
***         the HP-IB Interface Chip is off-line; however, Bit 12 = 0.

*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                        (HDADERR 5081)
***
***         HP-IB CONTROLLER bit (Bit 11) should be set because the device
***         adapter IS the System Controller and it has asserted the IFC line;
***         however, Bit 11 = 0.

*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                        (HDADERR 5082)
***
***         ADDRESSED TO TALK OR IDENTIFY bit (Bit 13) should be set because
***         the TALK ALWAYS bit is set (within the ADDRESS register);
***         however, Bit 13 = 0.

*** ERROR -- HP-IB INTERFACE CHIP FAILURE.                        (HDADERR 5083)
***
***         ADDRESSED TO LISTEN bit (Bit 14) should be set because the
***         LISTEN ALWAYS bit is set (within the ADDRESS register);
***         however, Bit 14 = 0.

*** ERROR -- INCONSISTENCY IN SYSTEM CONTROLLER STATUS.           (HDADERR 5084)
***
***         The SYSTEM CNTL bit within the HPIB_STATUS register
***         is reset, and the HP-IB Interface Chip is on-line;
***         however, Bit 12 = 1.


*** ERROR -- INCONSISTENCY IN SYSTEM CONTROLLER STATUS.           (HDADERR 5085)
***
***         The SYSTEM CNTL bit within the HPIB_STATUS register
***         is set; however, Bit 12 = 0.
```

HP-IB DAR Operations Used:

HPIB_LOCK
HPIB_READ_REG
HPIB_UNLOCK

Step 63—Read HP-IB Interface Chip CONTROL Register

This step will retrieve the contents of the frontplane interface chip's CONTROL Register and decode the respective control bits in order to display user-friendly messages for the user.

Note The frontplane interface chip's STATUS and ADDRESS registers will need to be accessed in order to fully decode the CONTROL register. This is because it must be known whether or not the device adapter is the System Controller (and whether the card is on-line or off-line) in order to decode the REN VALUE and IFC VALUE bits. The OUTBOUND DATA FREEZE bit in the STATUS register is needed to fully decode the implications of the RESPOND TO PARALLEL POLL bit.

Possible Output Messages:

```
Section 6 -- Status

   Step 63 - Read HP-IB Interface Chip CONTROL Register

      The OUTBOUND DATA FREEZE restriction is NOT in affect on the
        Parallel Poll response.                            (Bit  0 = 0)
      The OUTBOUND DATA FREEZE restriction IS in affect on the
        Parallel Poll response.                            (Bit  0 = 1)

      NDAC or NRFD signals are NOT delayed to the Source Handshake
        circuitry during DATA transfers.                   (Bit  1 = 0)
      NDAC or NRFD signals ARE delayed to the Source Handshake
        circuitry during DATA transfers.                   (Bit  1 = 1)

      Currently utilizing the standard 10-bit data path.   (Bit  8 = 0)
      Currently utilizing the 8-bit data path.             (Bit  8 = 1)

      Even parity Interface Commands ARE accepted.         (Bit  9 = 0)
      Even parity Interface Commands are NOT accepted.     (Bit  9 = 1)

      The REN line is at a logic LOW state.                (Bit 10 = 0)
      The local REN line is at a logic LOW state (the HP-IB
        Interface Chip is off-line).                       (Bit 10 = 0)
      The REN line is at a logic HIGH state.               (Bit 10 = 1)
      The local REN line is at a logic HIGH state (the HP-IB
        Interface Chip is off-line).                       (Bit 10 = 1)
      Decoding the REN VALUE bit is noninformative because the
        device adapter is not the System Controller.       (Bit 10)

      The IFC line is at a logic LOW state.                (Bit 11 = 0)
      The local IFC line is at a logic LOW state (the HP-IB
        Interface Chip is off-line).                       (Bit 11 = 0)
      The IFC line is at a logic HIGH state.               (Bit 11 = 1)
      The local IFC line is at a logic HIGH state (the HP-IB
        Interface Chip is off-line).                       (Bit 11 = 1)
```

```
Decoding the IFC VALUE bit is noninformative because the
  device adapter is not the System Controller.          (Bit 11)

The RESPOND TO PARALLEL POLL bit = 0; therefore the HP-IB
  Interface Chip is not in need of service.             (Bit 12 = 0)
The RESPOND TO PARALLEL POLL bit = 1; therefore the HP-IB
  Interface Chip will indicate the need for service during
  any Parallel Poll if it has the response capability. (Bit 12 = 1)
The RESPOND TO PARALLEL POLL bit = 1; however, since both
  the OUTBOUND DATA FREEZE and POLL HOLDOFF bits are set,
  the HP-IB Interface Chip will NOT respond affirmatively
  to a Parallel Poll.                                   (Bit 12 = 1)

The REQUEST SERVICE bit = 0; therefore the HP-IB
  Interface Chip will NOT request service during the
  next Serial Poll.                                     (Bit 13 = 0)
The REQUEST SERVICE bit = 1; depicting that the HP-IB
  Interface Chip has asserted the SRQ line and during the
  next Serial Poll will request service from the
  HP-IB Controller.                                     (Bit 13 = 1)

A DMA request will be issued when the Inbound FIFO is
  ready for a READ operation.                           (Bit 14 = 0)
A DMA request will be issued when the Outbound FIFO is
  ready for a WRITE operation.                          (Bit 14 = 1)

The INITIALIZE OUTBOUND FIFO bit is not architected to
  be read (0 is always returned).                       (Bit 15 = 0)

A binary representation of the respective register follows:
NOTE: Dashes represent undefined bits.

   ---------------------------------------------------------
   HP-IB Interface Chip CONTROL register's image:

   Bit #:     0     4     8     12    16    20    24    28

   Value:     ####  ####  ####  ####  ####  ####  ####  ####
   ---------------------------------------------------------

End of Step 63 - Read HP-IB Interface Chip CONTROL Register

End of Section 6 -- Status
```

Possible Error/Warning Messages:

```
*** WARNING -- The INITIALIZE OUTBOUND FIFO bit value was read in as
***            being 1, but should always be read as 0.        (HDADWARN 6030)
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_READ_REG
HPIB_UNLOCK
```

Step 64—Read HP-IB Interface Chip ADDRESS Register

When this step is run, HPIBDAD will read, decode, and display the contents of the HP-IB Interface Chip's ADDRESS register. The information maintained by this register pertains to the HP-IB address of the device adapter, as well as related control information.

Note In order to decode the full implications of the TALK ALWAYS and LISTEN ALWAYS bits, the frontplane interface chip's STATUS register must be accessed. Specifically, the ADDRESSED TO TALK and ADDRESSED TO LISTEN bits must be retrieved.

Possible Output Messages:

```
Section 6 -- Status

   Step 64 - Read HP-IB Interface Chip ADDRESS Register

      The CRC capabilities are DISABLED.                      (Bit  0 = 0)
      The CRC capabilities are ENABLED.                       (Bit  0 = 1)

      HP-IB commands originating from the Outbound FIFO will
        have ODD parity.                                      (Bit  1 = 0)
      HP-IB commands originating from the Outbound FIFO will
        have EVEN parity.                                     (Bit  1 = 1)

      The HP-IB Interface Chip is OFF-LINE.                   (Bit  8 = 0)
      The HP-IB Interface Chip is ON-LINE.                    (Bit  8 = 1)

      The TALK ALWAYS bit is RESET.                           (Bit  9 = 0)
      The TALK ALWAYS bit is reset; however, the device
        adapter IS addressed to Talk.                         (Bit  9 = 0)
      The TALK ALWAYS bit is SET.                             (Bit  9 = 1)

      The LISTEN ALWAYS bit is RESET.                         (Bit 10 = 0)
      The LISTEN ALWAYS bit is reset; however, the device
        adapter IS addressed to Listen.                       (Bit 10 = 0)
      The LISTEN ALWAYS bit is SET.                           (Bit 10 = 1)

      The HP-IB address for the device adapter
        is as follows (in decimal): !                         (Bits 11..15)
```

```
A binary representation of the respective register follows:
NOTE: Dashes represent undefined bits.

   ---------------------------------------------------------
   ADDRESS register's image:

   Bit #:     0     4     8     12    16    20    24    28

   Value:     ####  ####  ####  ####  ####  ####  ####  ####
   ---------------------------------------------------------

End of Step 64 - Read HP-IB Interface Chip ADDRESS Register

End of Section 6 -- Status
```

Possible Error/Warning Messages:

None specified by the diagnostic.

HP-IB DAR Operations Used:

HPIB_LOCK
HPIB_READ_REG
HPIB_UNLOCK

Step 65—Read HP-IB Interface Chip PP/ID_BYTE Registers

Executing this step will result in HPIBDAD displaying the contents of the PARALLEL_POLL_MASK/FIRST_ID_BYTE and PARALLEL_POLL_SENSE/SECOND_ID_BYTE registers to the diagnostician. The respective register's contents will not be decoded and displayed in a user-friendly fashion for the user in this step, but rather simply displayed as a binary bit pattern. The reason being that decoding the bits into user-friendly messages would actually be encoding the bits into "user-frustrating" messages. The double role of the two registers also adds to the argument for not decoding the bits, since the diagnostic cannot be sure of the registers' current role (i.e. which decode template to utilize when displaying the contents).

Possible Output Messages:

```
Section 6 -- Status

   Step 65 - Read HP-IB Interface Chip PP/ID_BYTE Registers

      A binary representation of the respective registers follows:
      NOTE: Dashes represent undefined bits.

         ----------------------------------------------------------
         PARALLEL_POLL_MASK/FIRST_ID_BYTE register's image:

         Bit #:     0     4     8     12    16    20    24    28

         Value:     ####  ####  ####  ####  ####  ####  ####  ####
         ----------------------------------------------------------
         PARALLEL_POLL_SENSE/SECOND_ID_BYTE register's image:

         Bit #:     0     4     8     12    16    20    24    28

         Value:     ####  ####  ####  ####  ####  ####  ####  ####
         ----------------------------------------------------------

   End of Step 65 - Read HP-IB Interface Chip PP/ID_BYTE Registers

End of Section 6 -- Status
```

Possible Error/Warning Messages:

None specified by the diagnostic.

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_READ_REG
HPIB_UNLOCK
```

Step 66—Read HPIB_STATUS Register

This step is used to obtain information about the current status (configuration) of the HP-IB and the HP-IB Device Adapter Talker/Listener/Controller Chip . This information is maintained in the read-only HPIB_STATUS register on the HP-IB DA. After this register is read in by the diagnostic program, the respective bits are decoded and displayed to the operator in a user-friendly manner.

Note In order to determine the consistency of reporting pending interrupts, the HP-IB Interface Chip's INTERRUPTING_CONDITIONS register will have to be read in.

Possible Output Messages:

```
Section 6 -- Status

   Step 66 - Read HPIB_STATUS Register

      HP-IB Interface Chip is NOT the System Controller.     (Bit 24 = 0)
      HP-IB Interface Chip IS the System Controller.         (Bit 24 = 1)

      HP-IB Interface Chip is in SLOW mode.                  (Bit 25 = 0)
      HP-IB Interface Chip is in FAST mode.                  (Bit 25 = 1)

      HP-IB Interface Chip's D1 value from the last read
        operation was 0.                                     (Bit 26 = 0)
      HP-IB Interface Chip's D1 value from the last read
        operation was 1.                                     (Bit 26 = 1)

      HP-IB Interface Chip's DO value from the last read
        operation was 0.                                     (Bit 27 = 0)
      HP-IB Interface Chip's DO value from the last read
        operation was 1.                                     (Bit 27 = 1)

      HP-IB Interface Chip is NOT interrupting.              (Bit 28 = 0)
      HP-IB Interface Chip IS interrupting.                  (Bit 28 = 1)

      Group Execute Trigger has NOT interrupted.             (Bit 29 = 0)
      Group Execute Trigger HAS interrupted.                 (Bit 29 = 1)

      Interface Clear has NOT interrupted.                   (Bit 30 = 0)
      Interface Clear HAS interrupted.                       (Bit 30 = 1)

      Hardware did NOT pass most recent test.                (Bit 31 = 0)
      Hardware PASSED most recent test.                      (Bit 31 = 1)


      A binary representation of the respective register follows:
      NOTE: Dashes represent undefined bits.
```

```
---------------------------------------------------------
HPIB_STATUS register's image:

Bit #:     0     4     8     12    16    20    24    28

Value:     ####  ####  ####  ####  ####  ####  ####  ####
---------------------------------------------------------
```

```
      End of Step 66 - Read HPIB_STATUS Register

   End of Section 6 -- Status
```

Possible Error/Warning Messages:

```
*** ERROR -- INCONSISTENCY IN INTERRUPT PENDING STATUS.        (HDADERR 5090)
***
***          INTERRUPTING_CONDITIONS register: bit  0 = 1
***          HPIB_STATUS register            : bit 28 = 0

*** ERROR -- INCONSISTENCY IN INTERRUPT PENDING STATUS.        (HDADERR 5091)
***
***          INTERRUPTING_CONDITIONS register: bit  0 = 0
***          HPIB_STATUS register            : bit 28 = 1
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_READ_REG
HPIB_UNLOCK
```

Step 67—Read BUS_STATUS Register

This step is used to obtain information about the current state of the HP-IB control bus (5 general interface management lines and 3 handshake lines) by reading in the contents of the BUS_STATUS Register. The image of the HP-IB control bus is latched into this register with the leading edge of the select signal for the register. The output of this step is the fully decoded representation of the HP-IB status.

Note Reading this register is an effective way to determine if there is any device in handshake mode connected to the HP-IB. The reason for this is that when a handshake device is in fact connected to the HP-IB, at least one of the following signals will be at the logic zero level: NDAC, NRFD. When there are no handshake devices connected to the HP-IB, these signals will both be at the logic one level.

Possible Output Messages:

```
Section 6 -- Status

   Step 67 - Read BUS_STATUS Register

      The logic levels of the respective bits are as follows:

                    Line ID                      Logic Level
                    -------                      -----------
       EOI - End or Identify     (Bit 24):            #
       REN - Remote Enable       (Bit 25):            #
       SRQ - Service Request     (Bit 26):            #
       ATN - Attention           (Bit 27):            #
       IFC - Interface Clear     (Bit 28):            #

       DAV - Data valid          (Bit 29):            #
      NDAC - Not Data Accepted   (Bit 30):            #
      NRFD - Not Ready For Data  (Bit 31):            #

      A binary representation of the respective register follows:
      NOTE: Dashes represent undefined bits.

       ---------------------------------------------------------
       BUS_STATUS register's image:

       Bit #:     0    4    8    12   16   20   24   28

       Value:     #### #### #### #### #### #### #### ####
       ---------------------------------------------------------


   End of Step 67 - Read BUS_STATUS Register

End of Section 6 -- Status
```

Possible Error/Warning Messages:

```
*** WARNING -- No device in handshake mode is connected to the HP-IB,
***            depicted by the NDAC and NRFD lines both being at the
***            logic one level.                                 (HDADWARN 6050)
```

HP-IB DAR Operations Used:

HPIB_LOCK
HPIB_READ_REG
HPIB_UNLOCK

Section 10—Register Level Input/Output Transactions

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

Warning **The user of this section must exercise EXTREME CAUTION when sending information over the HP-IB. If a negligent user sends "garbage" information to a system disk, the result could be a corrupted disk and a system crash!**

This section can be executed by the diagnostician when he/she wishes to do "peeks" and/or "pokes" to the device adapter. That is to say, this section allows the user to read/write any register in the device adapter's address space.

Warning **In order for this section to be of any value, the device adapter being diagnosed should not be accessed by any process other than** `HPIBDAD`**. This is to ensure that any configuration modifications made by the diagnostic are pending for succeeding transactions.**

Possible Output Messages:

```
Section 10 -- Register Level Input/Output Transactions

   At the prompt, enter one of the following commands:
     1) "Input Register" Command:

             i <decimal register number>

        For Example:
          To input register number 159, type "i 159<CR>"

     2) "Output Register" Command:

             o <decimal register number> <hex data>

        For Example:
          To output 0x02BAD to register number 12 (decimal), type
          "o 12 2bad<CR>"

     3) A lone <CR> will cause this section to terminate.

  %

    : !

End of Section 10 -- Register Level Input/Output Transactions
```

Note All commands/data can be entered in upper and/or lower case characters.

Possible Error/Warning Messages:

```
*** WARNING -- This section is intended for personnel which have a good
***            understanding of the HP-IB Device Adapter's architecture.
***            It is strongly recommended that you print off "More Help" on
***            Section 10, or reference the user's manual before using this
***            section.                                        (HDADWARN 6060)

*** WARNING -- Invalid or missing register number ... the register number
***            value is not a properly formatted DECIMAL integer.
***            Please try again.                               (HDADWARN 6061)

*** WARNING -- Invalid register number ... the register number entered
***            is beyond the device adapter's address space (highest
***            register number is decimal 1023).
***            Please try again.                               (HDADWARN 6062)

*** WARNING -- Invalid or missing datum ... the datum value is not a properly
***            formatted HEXADECIMAL integer.
***            Please try again.                               (HDADWARN 6064)

*** WARNING -- Invalid command ... the command entered cannot be decoded.
***            Please try again.                               (HDADWARN 6065)
```

HP-IB DAR Operations Used:

HPIB_LOCK
HPIB_WRITE_REG
HPIB_READ_REG
HPIB_UNLOCK

Section 11—Data/Command Transaction on HP-IB

```
Minimum Mode Required         : Destructive

Terminal Used for Execution : Any

In Default Set?               : No

In Auto-diagnostic Set?       : No
```

Executing this section will allow the user to send data/commands to a particular device connected to the interface bus, as well as receive data from the device. Essentially, this section gives the user complete control of the information to be sent to the frontplane interface chip's Register 2 (Outbound FIFO). This implies that the user must have a very good working knowledge of the HP-IB protocol; however, examples of some of the more common transactions can be observed by running Section 1 (**More Help**) of this diagnostic and entering "11" at the prompt.

When this section is run, data entered is expected to be in hexadecimal format. Also, the user may restart information entry by entering a lone semicolon (;) at any prompt. Entering a lone <CR> will terminate information entry, whereas entering a lone colon (:) terminates data entry AND conveys to the diagnostic that return data is expected from a device on the HP-IB. When expected data is returned within the timeout period, it will be displayed in hexadecimal format; however, if the timeout period elapses, an appropriate error message will be output.

In order to make this section a little more beneficial to the user, the `HPIBDAD` will take care of issuing the necessary (Un)Talk/(Un)Listen addressing. Therefore, the user will be required to simply enter the address of the device he/she wishes to communicate with in addition to the data to be sent.

The user should make sure that if a command is being sent to a device, that it is "terminated" according to the protocol of the particular device. For example, some devices accept a semicolon (;) as a command delimiter, others expect the EOI line to be asserted, etc. (to assert the EOI line, the number two (2) should be inserted as the most significant digit into the byte where the EOI is desired; e.g. if an EOI is desired while sending the hex byte 49, the user should enter the hex datum of 249—this sets the END bit in the frontplane interface chip's Outbound FIFO).

Note

In order for this diagnostic section to execute properly, the HP-IB Device Adapter being diagnosed must be the System Controller. Therefore, the HPIB_STATUS register will be read in at the onset of this section in order to verify the System Controller status. If the device adapter under test is not the System Controller, a warning message will be output and the section will be exited. However, it is not expected that this scenario will often occur, since the device adapter under test will usually be the System Controller.

Also note that the REN (Remote Enable) Line will be asserted on the DA under test by `HPIBDAD` before the data/command transaction takes place on the HP-IB.

Caution

A timeout error condition may exist if the device adapter is not able to send data/commands over the HP-IB due to a "configuration" problem (e.g. DA is neither the HP-IB Controller nor addressed to talk). Therefore, in this type of situation, there actually isn't a hardware/software problem, but the user may infer this from the error message. Similarly, the user may have requested information to be returned from a device on the HP-IB, but that device may be very slow in collecting the requisite data, therefore causing an error message to be output when an actual hardware/software error may not exist.

Warning

The user of this section must exercise EXTREME CAUTION when sending information over the HP-IB. If a negligent user sends "garbage" information to a system disk, the result could be a corrupted disk and a system crash!

Possible Output Messages:

```
Section 11 -- Data/Command Transaction on HP-IB

   Example scripts can be observed by executing Section 1 -- More Help
   and entering "11" at the prompt.

   Please enter the HP-IB address of the device you wish to
   transact with (0..29) :

   At a prompt, enter one of the following:
     1) A hexadecimal integer (10-bit maximum per line--i.e. 0..3FF)
     2) A lone <CR> to terminate information entry
     3) A lone colon (:) to terminate information entry AND convey to the
        diagnostic that return data is expected
     4) A lone semicolon (;) to abort the current information entry session
        and to start over

   CAUTION: Care must be taken not to enter a hexadecimal integer that
   matches a language localized control message.  For example, if a lone
   'e' is entered, it is an indication to the diagnostic to exit.  In this
   case, "0e" should be entered.

   >>

   The device RETURNED the following data (displayed in hex format):

   ... Information entry to be restarted.

   Information being sent to HP-IB Interface Chip's Outbound FIFO ...

End of Section 11 -- Data/Command Transaction on HP-IB
```

Possible Error/Warning Messages:

```
*** ERROR -- THE NUMBER OF DATA BYTES RETURNED BY THE DEVICE EXCEEDED
***         THE SPECIFIED BYTE COUNT.                          (HDADERR 5100)
***
***         Byte count was set to: !
***         Data  bytes  returned: !

*** WARNING -- Invalid response.  Please answer the question with one
***            of the choices given.                           (HDADWARN 6001)

*** WARNING -- It is strongly recommended that you print off "More Help" on
***            Section 11, or reference the user's manual before using this
***            section.                                        (HDADWARN 6070)

*** WARNING -- The device adapter under test is NOT the System
***            Controller, therefore this section cannot be
***            executed.                                       (HDADWARN 6072)

*** WARNING -- Information entered is not a properly formatted
***            HEXADECIMAL integer.
***            Please try again.  For example: "2BE<CR>".      (HDADWARN 6075)

*** WARNING -- Information entered exceeds the 10-bit limit.
***            Please try again.  For example: "3FF<CR>".      (HDADWARN 6076)

*** WARNING -- The number of data units entered has exceeded the buffer size.
***            Please terminate the current data entry
***            when prompted.                                  (HDADWARN 6077)
```

HP-IB DAR Operations Used:

```
HPIB_LOCK
HPIB_WRITE_REG
HPIB_READ_REG
HPIB_INPUT
HPIB_RESET
HPIB_UNLOCK
```

Error and Warning Messages

The following provides a listing, in numerical order, of the most significant error and warning messages displayed by the system.

Error Messages

The following provides a listing, in numerical order, of the most significant error messages displayed by the system.

```
5005 *** ERROR -- IODC CHECKSUM FAILED.                              (HDADERR 5005)
```

```
5010 *** ERROR -- PB INTERFACE CHIP (PBIC) LOOPBACK FAILED.          (HDADERR 5010)
     ***
     ***        Data written to PBIC (in hex)      Data read from PBIC (in hex)
     ***        -----------------------------      ----------------------------
     ***                   !                                  !
```

```
5020 *** ERROR -- HP-IB INTERFACE CHIP (HIC) LOOPBACK FAILED.        (HDADERR 5020)
     ***
     ***        Data written to HIC (in hex)       Data read from HIC (in hex)
     ***        ----------------------------       ---------------------------
     ***                   !                                  !
```

```
5030 *** ERROR -- PB INTERFACE CHIP (PBIC) TEST FAILED.              (HDADERR 5030)
     ***        ! register failed the "Register Verification" test.
     ***
     ***        Data written to PBIC (in hex)      Data read from PBIC (in hex)
     ***        -----------------------------      ----------------------------
```

```
5031 *** ERROR -- HP-IB INTERFACE CHIP (HIC) TEST FAILED.          (HDADERR 5031)
     ***         ! register failed the
     ***         "Register Verification" test.
     ***
     ***         Data written to HIC (in hex)       Data read from HIC (in hex)
     ***         ----------------------------       ---------------------------
```

```
5035 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5035)
     ***         The interface chip "IFC" test failed.
```

```
5036 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5036)
     ***         The interface chip "Talk/Listen" test failed.
```

```
5037 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5037)
     ***         The interface chip "REN" test failed.
```

```
5038 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5038)
     ***         The interface chip "FIFO" test failed.
```

```
5039 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5039)
     ***         The interface chip "SRQ" test failed.
```

```
5040 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5040)
     ***         The interface chip "Parallel Poll" test failed.
```

```
5041 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.                (HDADERR 5041)
     ***         The interface chip "Secondary Address" test failed.
```

```
5042 *** ERROR -- HP-IB INTERFACE CHIP TEST FAILED.               (HDADERR 5042)
     ***          The interface chip "CRC" test failed.
```

```
5045 *** ERROR -- HPIB_CONTROL/STATUS REGISTER SET VERIFICATION
     ***          TEST FAILED.                                    (HDADERR 5045)
```

```
5046 *** ERROR -- GROUP EXECUTE TRIGGER OR HP-IB INTERFACE CHIP'S INTERRUPT
     ***          CIRCUITRY FAILED TEST.                          (HDADERR 5046)
```

```
5047 *** ERROR -- ON-LINE TEST FAILED.                            (HDADERR 5047)
     ***
     ***          This test is targeted to check the frontplane's General Interface
     ***          Management Lines (control transceiver), and BUS_STATUS register.
```

```
5048 *** ERROR -- IFC INTERRUPT TEST FAILED.                      (HDADERR 5048)
     ***
     ***          This test is targeted to check the frontplane's IFC
     ***          interrupt circuitry.
```

```
5049 *** ERROR -- DMA TEST NUMBER ! FAILED.                       (HDADERR 5049)
     ***          Data was corrupted.
```

```
5050 *** ERROR -- DMA TEST NUMBER ! FAILED.                       (HDADERR 5050)
     ***          DMA termination condition incorrect.
```

```
5051 *** ERROR -- DMA TEST NUMBER ! FAILED.                       (HDADERR 5051)
     ***          Incorrect residue value in the IO_DMA_COUNT register.
```

```
5052 *** ERROR -- DMA TEST NUMBER ! FAILED.                         (HDADERR 5052)
     ***          Internal Software Error.
```

```
5053 *** ERROR -- DMA TEST NUMBER ! FAILED.                         (HDADERR 5053)
     ***          Inconsistency within PB Interface Chip with respect to
     ***          reporting "DMA length conflict" status.
```

```
5060 *** ERROR -- PB INTERFACE CHIP FAILURE.                        (HDADERR 5060)
     ***
     ***          PB Interface Chip reported to have ! bytes in FIFO,
     ***          however only 24 bytes are available.
```

```
5061 *** ERROR -- PB INTERFACE CHIP FAILURE.                        (HDADERR 5061)
     ***
     ***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) is set
     ***          (implying FIFO CNT should = 24); however, the FIFO CNT field
     ***          (Bits 23..27) = ! (decimal).
```

```
5062 *** ERROR -- PB INTERFACE CHIP FAILURE.                        (HDADERR 5062)
     ***
     ***          DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) is set
     ***          (implying FIFO CNT should = 0); however, the FIFO CNT field
     ***          (Bits 23..27) = ! (decimal).
```

```
5063 *** ERROR -- PB INTERFACE CHIP FAILURE.                        (HDADERR 5063)
     ***
     ***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) and
     ***          FIFO EMPTY bit (Bit 21) are both set.
```

```
5064 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                    (HDADERR 5064)
     ***
     ***         THE "INTERRUPT PENDING" BIT WAS INADVERTENTLY SET:
     ***           The INTERRUPT PENDING bit (Bit 0) of
     ***           INTERRUPTING_CONDITIONS = 1; however, interrupts are
     ***           disabled, depicted by the INTERRUPT ENABLE bit (Bit 0) of
     ***           INTERRUPT_MASK = 0.
```

```
5065 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                    (HDADERR 5065)
     ***
     ***         AN INTERRUPTING CONDITION BIT WAS INADVERTENTLY SET:
     ***          One or more bits of INTERRUPTING_CONDITIONS = 1, even though the
     ***          interrupts have been masked, depicted by the respective bits of
     ***          INTERRUPT_MASK = 0.
```

```
5066 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                    (HDADERR 5066)
     ***
     ***         THE "INTERRUPT PENDING" BIT FAILED TO BE SET:
     ***           One or more bits of INTERRUPTING_CONDITIONS and Bit 0 of
     ***           INTERRUPTING_MASK = 1; however, the INTERRUPT PENDING bit
     ***           (Bit 0) of INTERRUPTING_CONDITIONS = 0.
```

```
5067 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                    (HDADERR 5067)
     ***
     ***         THE "INTERRUPT PENDING" BIT WAS INADVERTENTLY SET:
     ***           No interrupting condition bit is set, however the
     ***           INTERRUPT PENDING bit (Bit 0) of INTERRUPT_CONDITIONS = 1.
```

```
5068 *** ERROR -- INCONSISTENCY IN INTERRUPT STATUS.              (HDADERR 5068)
     ***
     ***         The "INTERRUPT PENDING" bit (Bit 0) of
     ***         INTERRUPTING_CONDITIONS = 1;
     ***         however, the interrupt input bit of the DIAGNOSTIC_STATUS
     ***         register (Bit 0) = 1, which indicates that the PB Interface
     ***         Chip did not recognize the interrupt request.
```

```
5069 *** ERROR -- PB INTERFACE CHIP FAILURE.                      (HDADERR 5069)
     ***
     ***         DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) = 0
     ***         (implying FIFO CNT should be less than 24); however, the
     ***         FIFO CNT field (Bits 23..27) = ! (decimal).
```

```
5070 *** ERROR -- PB INTERFACE CHIP FAILURE.                      (HDADERR 5070)
     ***
     ***         DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) = 0
     ***         (implying FIFO CNT should be greater than 0); however, the
     ***         FIFO CNT field (Bits 23..27) = ! (decimal).
```

```
5080 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                   (HDADERR 5080)
     ***
     ***         HP-IB SYSTEM CONTROLLER bit (Bit 12) should be set because
     ***         the HP-IB Interface Chip is off-line; however, Bit 12 = 0.
```

```
5081 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                    (HDADERR 5081)
     ***
     ***          HP-IB CONTROLLER bit (Bit 11) should be set because the device
     ***          adapter IS the System Controller and it has asserted the IFC line;
     ***          however, Bit 11 = 0.
```

```
5082 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                    (HDADERR 5082)
     ***
     ***          ADDRESSED TO TALK OR IDENTIFY bit (Bit 13) should be set because
     ***          the TALK ALWAYS bit is set (within the ADDRESS register);
     ***          however, Bit 13 = 0.
```

```
5083 *** ERROR -- HP-IB INTERFACE CHIP FAILURE.                    (HDADERR 5083)
     ***
     ***          ADDRESSED TO LISTEN bit (Bit 14) should be set because the
     ***          LISTEN ALWAYS bit is set (within the ADDRESS register);
     ***          however, Bit 14 = 0.
```

```
5084 *** ERROR -- INCONSISTENCY IN SYSTEM CONTROLLER STATUS.        (HDADERR 5084)
     ***
     ***          The SYSTEM CNTL bit within the HPIB_STATUS register
     ***          is reset, and the HP-IB Interface Chip is on-line;
     ***          however, Bit 12 = 1.
```

```
5085 *** ERROR -- INCONSISTENCY IN SYSTEM CONTROLLER STATUS.        (HDADERR 5085)
     ***
     ***         The SYSTEM CNTL bit within the HPIB_STATUS register
     ***         is set; however, Bit 12 = 0.
```

```
5090 *** ERROR -- INCONSISTENCY IN INTERRUPT PENDING STATUS.        (HDADERR 5090)
     ***
     ***           INTERRUPTING_CONDITIONS register: bit  0 = 1
     ***           HPIB_STATUS register            : bit 28 = 0
```

```
5091 *** ERROR -- INCONSISTENCY IN INTERRUPT PENDING STATUS.        (HDADERR 5091)
     ***
     ***           INTERRUPTING_CONDITIONS register: bit  0 = 0
     ***           HPIB_STATUS register            : bit 28 = 1
```

```
5100 *** ERROR -- THE NUMBER OF DATA BYTES RETURNED BY THE DEVICE EXCEEDED
     ***         THE SPECIFIED BYTE COUNT.                          (HDADERR 5100)
     ***
     ***         Byte count was set to: !
     ***         Data  bytes  returned: !
```

Warning Messages

The following provides a listing, in numerical order, of the most significant warning messages displayed by the system.

6001

```
*** WARNING -- Invalid response.  Please answer the question with one
***            of the choices given.
(HDADWARN 6001)
More Help (1..6, 10, 11, <CR>) :
```

6010

```
*** WARNING -- Soft Physical Address Capability = 0x!,

***            I expected  0x80.
(HDADWARN 6010)
```

6011

```
*** WARNING -- Type of Module = ! (UNKNOWN PRODUCT),
***            I expected a 4 (TP_A_DMA).
(HDADWARN 6011)
```

6015

```
*** WARNING -- The device adapter under test is NOT the System

***            Controller, therefore the "On-Line Test" cannot be

***            executed.
(HDADWARN 6015)
```

```
6016    *** WARNING -- The device adapter under test is NOT the System
        ***            Controller, therefore the "IFC Interrupt Test" cannot

        ***            be executed.
        (HDADWARN 6016)
```

```
6020    *** WARNING -- The IO_STATUS register has one or more undefined bits
        ***            reading in as a 1 (I expected 0's).
        (HDADWARN 6020)
```

```
6022    *** WARNING -- A SOFT ERROR (length conflict) occurred since the last
        ***            write to the IO_COMMAND register.

        ***            (IO_STATUS Bit 22 = 1)
        (HDADWARN 6022)
```

```
6023    *** WARNING -- A FATAL ERROR occurred since the last write to the
        ***            IO_COMMAND register.

        ***            (IO_STATUS Bit 24 = 1)
        (HDADWARN 6023)
```

```
6025    *** WARNING -- PB Interface Chip HAS generated a PB bus error since
        ***            the last reset command.

        ***            (DIAGNOSTIC_STATUS Bit 1 = 0)
        (HDADWARN 6025)
```

6030 *** WARNING -- The INITIALIZE OUTBOUND FIFO bit value was read in as
*** being 1, but should always be read as 0.
(HDADWARN 6030)

6050 *** WARNING -- No device in handshake mode is connected to the HP-IB,
*** depicted by the NDAC and NRFD lines both being at the
*** logic one level.
(HDADWARN 6050)

6060 *** WARNING -- This section is intended for personnel which have a good
*** understanding of the HP-IB Device Adapter's architecture.
*** It is strongly recommended that you print off "More Help"
on
*** Section 10, or reference the user's manual before using
this
*** section.
(HDADWARN 6060)

6061 *** WARNING -- Invalid or missing register number ... the register
number
*** value is not a properly formatted DECIMAL integer.
*** Please try again.
(HDADWARN 6061)

6062 *** WARNING -- Invalid register number ... the register number entered
*** is beyond the device adapter's address space (highest
*** register number is decimal 1023).
*** Please try again.
(HDADWARN 6062)

6064 *** WARNING -- Invalid or missing datum ... the datum value is not a properly

*** formatted HEXADECIMAL integer.

*** Please try again.
(HDADWARN 6064)

6065 *** WARNING -- Invalid command ... the command entered cannot be decoded.

*** Please try again.
(HDADWARN 6065)

6070 *** WARNING -- It is strongly recommended that you print off "More Help" on

*** Section 11, or reference the user's manual before using this

*** section.
(HDADWARN 6070)

6072 *** WARNING -- The device adapter under test is NOT the System

*** Controller, therefore this section cannot be

*** executed.
(HDADWARN 6072)

6075 *** WARNING -- Information entered is not a properly formatted

*** HEXADECIMAL integer.

*** Please try again. For example: "2BE<CR>".
(HDADWARN 6075)

6076 *** WARNING -- Information entered exceeds the 10-bit limit.

*** Please try again. For example: "3FF<CR>".
(HDADWARN 6076)

6077 *** WARNING -- The number of data units entered has exceeded the buffer size.

*** Please terminate the current data entry

*** when prompted.
(HDADWARN 6077)

6

GPIO Device Adapter Diagnostic

Introduction

The GPIO Device Adapter Diagnostic (GPIODAD) is part of the online diagnostics package. It is designed to provide its user with a means of determining if the specified GPIO Device Adapter (DA) and its related hardware are operating properly, and if not, which FRU should be replaced. There are a variety of tests which the user can run to determine the source of a problem. Some of the tests require writing and reading data to and from (respectively) the DA after a loopback hood has been placed on the DA's frontplane.

GPIODAD can be utilized in three ways: 1) to verify the functionality of the GPIO Device Adapter (part number 28651A), 2) to isolate a fault within the GPIO interface down to the Field Replaceable Unit (FRU) level, and 3) to isolate a fault within the GPIO Device Adapter down to the component level (if possible). Additional features of GPIODAD allow the user to reset the GPIO Device Adapter and read/write various registers.

In order to execute a large percentage of GPIODAD, the user must have the capability to run destructive tests. However, some of the sections do allow the user to execute them in Normal Mode. All of the tests can be run from any terminal.

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10029.

Minimum Configuration

The hardware required to run GPIODAD consists of a Precision Bus (PB) GPIO Device Adapter (part number 28651A). as well as an 815 computer system with all the necessary equipment to bring the operating system (HP-UX) up and running.

Moreover, in order to *fully* test the GPIO Device Adapter, a GPIO loopback hood (part number 28651-60003) must be connected to the GPIO. This will allow the frontplane drivers/receivers and external interrupt circuitry to be tested.

In order to run GPIODAD, the online diagnostics diagnostic system must be present, along with the portability interface routines. Also, the diagnostic must guarantee that regardless of what condition the GPIO Device Adapter (DA) to be tested is in, the system will not crash by running the diagnostic on it.

Other software that must be present in the system to execute GPIODAD includes the following: the GPIO Device Access Routines (DAR) which act as an interface between the diagnostics and the driver, and the GPIO Device Adapter Manager (DAM) which is the driver to the GPIO DA. Note that the DAR is technically part of the portability interface, as far as the online diagnostics module structure is concerned.

Operating Instructions

GPIODAD can be run in two different modes, which are described below. Nevertheless, the description of all three diagnostic user environments (with respect to MPE/iX/online diagnostics) are illustrated for the sake of completeness.

Note

GPIODAD does not execute any tests in *Disruptive Mode*, nor is this mode supported by HP-UX/online diagnostics. Within HP-UX/online diagnostics, any sections/steps that are designated to be executed in *Disruptive Mode* are executed in *Normal Mode* (see below).

- *Disruptive Mode* - Indicates that the program can run tests on the selected device that are disruptive in nature. A disruptive test is one that does not destroy any data on the device, but could cause errors for other users on the system. For example, if a user was to run the internal selftest on a system disk, that test might be considered disruptive since the disk would temporarily go off-line to perform the test, thus causing errors for others who tried to access the disk at that time.
- *Destructive Mode* - Indicates that the program may run any test it desires on the selected device. This mode is required for tests that have the potential for corrupting data on the device being tested (i.e., Destructive tests). There are virtually no restrictions on tests run in this mode; therefore, this mode is handled with extreme care by the diagnostic program. An example of a test that would require this mode in order to run would be one that reformats the media on a system disk, thus destroying all of the data on it.
- *Normal Mode* - Indicates that the diagnostic program cannot run any tests on the selected device that are considered to be potentially destructive or disruptive in nature.

GPIODAD can be accessed by users via the Diagnostic User Interface (DUI) provided by the online diagnostics.

Default Tests

The following are the default tests and sections for GPIODAD.

```
Section 3     Identify

Section 6     Status

  Step 61       Read PB Interface Chip IO_STATUS Register
```

RUN Command

To bring up the Online Diagnostic subsystem, enter the following command to the HP-UX system prompt:

```
% sysdiag
```

The diagnostic subsystem responds with the following prompt indicating that access has been granted to the user:

```
DUI >
```

Typing `HELP` causes a summary of the DUI function and its commands to appear on the screen.

Note The device to be tested must be powered up and on line. The physical device location (pdev) shown below matches the same device shown on the "typical A1002A" system configuration, described in the chapter on DUI. The pdev value entered must be correct for the system being tested.

`GPIODAD` is initiated by using the **run gpiodad** command. For further information on the **run** command provided by the DUI, please refer to the online diagnostics DUI ES. All parameters available in the **run** command are acceptable as parameters when executing this diagnostic.

Note All of the sections in this diagnostic can be executed from any terminal, even if a specific test requires the user to have the capability to run destructive tests. This implementation therefore allows the diagnostic to be run from a remote terminal; however, there may be system limitations that would not allow the use of a remote terminal.

Test Execution

When GPIODAD is invoked, a banner and welcome message will be displayed.

After the banner and welcome message are displayed via a program services call, the diagnostic will call another program services routine in order to obtain access to the device that was selected for testing (in addition to setting up the sections and steps to be run).

This routine will exit with its *status* parameter (passed by reference) being any one of three possible values. The first of which is *successful*. This indicates that all sections and steps have been validated and that the system granted access to the device.

The second possible value is *dssd_device_in_use*. If this value is returned, it indicates that the system did not grant access to the device. If this happens, the following error message will be issued by the diagnostic:

```
*** ERROR -- GPIO DEVICE ADAPTER ALREADY IN USE BY THE
***          DIAGNOSTIC SYSTEM.                               (GDADERR 5000)
***
***          Someone has already gained exclusive rights to the
***          GPIO Device Adapter that you requested, and it is illegal
***          to have two copies of the GPIODAD diagnosing the same GPIO
***          Device Adapter simultaneously.
```

The diagnostic will terminate execution after outputting this error message.

The third possible status value is *dssd_internal_error*. When this value is active upon exiting the subroutine, it indicates that an error such as no device adapter at the specified LDEV was found. online diagnostics itself will output the error message for this situation, *not* GPIODAD. The diagnostic will terminate upon regaining control.

If all went well up to this point, the sections and steps specified by the user will be executed and the results displayed. If the user did not specify any sections/steps to be run, the default sections and steps will be executed (Sections 3 and 6). If at any time, the number of errors generated exceeds the limit specified by the user in the *errcount* parameter (of the DUI **run** command), the following message will be output:

```
*** WARNING -- The maximum specified number of error occurrences has
***            been exceeded.                                    (GDADWARN 6000)
```

The diagnostic will then terminate its execution. If the *errpause* parameter of the **run** command was assigned to "on", then the diagnostic will stop after each error is generated and ask the user if the testing should continue. The prompt that will be displayed is as follows:

```
Do you wish to continue? (Y/N) [Y] :
```

If the response is "Y", then the testing will resume (if possible), and if the response is "N", the diagnostic will terminate its execution. If the sections and steps specified by the user were executed the number of times specified in the *loop* parameter of the **run** command without the number of errors exceeding the *errcount* value, the diagnostic will terminate normally.

At any time that the diagnostic is prompting the user for information, the user may enter "exit" to terminate its execution, or enter "suspend" to temporarily suspend its execution. Either the entire word or any number of characters which uniquely identify the respective language localized command may be entered. Moreover, the letters entered may be in any combination of upper and lower case characters. If the user exits in this fashion, the following message is displayed:

```
... Exiting GPIODAD per your request.
```

If the user temporarily suspends execution in this manner, the message that will be displayed is as follows:

```
... GPIODAD suspended per your request.
```

The user can then perform tasks through the Diagnostic User Interface (DUI) and subsequently resume execution of GPIODAD, or he/she can abort the GPIODAD entirely.

Detailed Test Descriptions

The remainder of this section discusses each section and step in detail. As a quick reference, the following table was included to list all of the sections and steps available for use in GPIODAD.

```
  Section No.       Diagnostic Function
===============================================================================

       1            More Help

       2            Reset

       3            Identify

       4            Local Loopback from PB Interface Chip

       5            Hardware Test

       6            Status
                       Step  61 - Read PB Interface Chip IO_STATUS Register
                       Step  62 - Read PB Interface Chip
                                    DIAGNOSTIC_STATUS Register
                       Step  63 - Read Device Adapter IO_GPIO_STATUS Register
                       Step  64 - Read Device Adapter IO_GPIO_INHI/LO Registers

       8            External Loopback

      10            Register Level Input/Output Transactions

      11            Data Transaction on GPIO
```

Section 1 **More Help** [*Normal Mode, Not in Default Set*] Gives additional information about each of the sections and steps.

Section 2 **Reset** [*Destructive Mode, Not in Default Set*] This section initiates the following activities: reset the GPIO Device Adapter (DA) and the GPIO Device Adapter Manager (DAM) to a known state, load the appropriate information into the DA in order to have it operate properly.

Section 3 **Identify** [*Destructive Mode, In Default Set*] Issues an `Identify` command to the GPIO DAM, which acquires nearly all of the requested information from the GPIO DA's IODC. The information returned to the diagnostic is then decoded and displayed.

Section 4 **Local Loopback from PB Interface Chip** [*Destructive Mode, Not in Default Set*] The local loopback tests will determine the operational status of the backplane of the GPIO DA.

Section 5 **Hardware Test** [*Destructive Mode, Not in Default Set*] This section will perform a thorough hardware test on the GPIO DA. If the tests determine that something within the DA is abnormal, messages indicating what the problem is likely to be are displayed to the user.

Section 6 **Status** [*Destructive Mode, In Default Set*] This section is broken down into four steps so that the diagnostician can convey to the GPIODAD precisely what status information he/she wishes to examine. Moreover, the diagnostician can have a very thorough status report of the GPIO DA if he/she desires it by executing all four steps.

Step 61 **Read PB Interface Chip IO_STATUS Register**

Step 62 **Read PB Interface Chip DIAGNOSTIC_STATUS Register**

Step 63 **Read Device Adapter IO_GPIO_STATUS Register**

Step 64 **Read Device Adapter IO_GPIO_INHI/LO Registers**

Section 8 **External Loopback** [*Destructive Mode, Not in Default Set*] This section performs a loopback test on the entire data path of the GPIO DA, including the frontplane drivers/receivers.

Section 10 **Register Level Input/Output Transactions** [*Destructive Mode, Not in Default Set*] This section can be executed by the user to do peeks or pokes to any address on the GPIO DA.

Section 11 **Data Transaction on GPIO**

Section 1—More Help

```
Minimum Mode Required         : Normal

Terminal Used for Execution : Any

In Default Set?               : No

In Auto-diagnostic Set?       : No
```

More Help is an interactive section which allows the user to obtain more information about a particular section than is given when typing **help gpiodad** at the DUI prompt. This is needed because it is not desirable to spew large help screens at the user when he/she is looking for general help, but it is desirable to give more information about certain sections when requested.

This section allows all users from any terminal to obtain the additional information that they request.

Possible Output Messages:

```
Section 1 -- More Help

   This Section allows you to get more information on any of the Sections
   [1..6, 8, 10, 11] of this diagnostic.  Please indicate the number of the
   section for which you require more information.  Entering a lone <CR> to
   the prompt exits this section.

   More Help (1..6, 8, 10, 11, <CR>) :


End of Section 1 -- More Help
```

If the user enters a section number in response to the prompt, the pertinent information would be displayed for the user to read. The numbering scheme used for the help messages is as follows: the number of the first message of each section is equal to *(section_number) * 100 + 10000*. This allows for 100 messages per section. Note that when multiple messages exist for a given section, all of the corresponding messages are displayed.

Possible Error/Warning Messages:

If the user inputs a number of a section that is not implemented (i.e., not in the set {1,2,3,4,5,6,8,10,11}), then the following message is output:

```
*** WARNING -- Invalid response.  Please answer the question with one
***            of the choices given.                          (GDADWARN 6001)

More Help (1..6, 8, 10, 11, <CR>) :
```

Note that the user is prompted again for input.

GPIO DAR Operations Used:

None

Section 2—Reset

```
Minimum Mode Required         : Destructive

Terminal Used for Execution : Any

In Default Set?               : No

In Auto-diagnostic Set?       : No
```

This section informs the Device Adapter Manager (DAM)—via the Device Access Routines—to reset the GPIO Device Adapter (DA) and DAM to its power-on state. The DAM will then transfer all pertinent data onto the DA that the DA needs to operate properly.

Possible Output Messages:

```
Section 2 -- Reset

   NO ERRORS DETECTED while resetting the device adapter.

End of Section 2 -- Reset
```

Possible Error/Warning Messages:

```
None specified by the diagnostic.
```

GPIO DAR Operations Used:

```
gpio_reset_card
```

Section 3—Identify

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : Yes

In Auto-diagnostic Set?      : No
```

Identify issues an `Identify` command to the GPIO DAR, which acquires the requested information from the GPIO DAM, which in turn acquires most of the information from the GPIO DA's IODC (the only information not stored in the IODC is the DAM's version code). The diagnostic then decodes the information obtained and displays it in a manner that is informative to the user. This section can be used to determine the GPIO DA's hardware and software versions, as well as the version of the DAM being used. This section has an added benefit in that if it executes successfully, the path from the diagnostic to the GPIO DA is known to be at least partially functional.

Note Hardware Version and Software Version will be displayed as hexadecimal numbers.

Possible Output Messages:

```
Section 3 -- Identify


   Hardware Version                 : ?

   Soft Physical Address Capability : 0

   Type of Module                   : 4 (Type A DMA I/O Adapter)

   Software Version                 : ?

   Device Adapter Manager Version   : ?

   DAR Version                      : ?

End of Section 3 -- Identify
```

Note The Hardware Version, Software Version, and Device Adapter Manager Version fields may vary in time, therefore cannot be explicitly specified within this document.

Possible Error/Warning Messages:

If the Soft Physical Address Capability is not 0, the following message is displayed:

```
*** WARNING -- Soft Physical Address Capability = !,
***            I expected a 0.                              (GDADWARN 6010)
```

If the module type value returned is not that of a Type A DMA I/O Adapter, the following message is displayed:

```
*** WARNING -- Type of Module = ! (UNKNOWN PRODUCT),
***            I expected a 4 (TP_A_DMA).                   (GDADWARN 6011)
```

GPIO DAR Operations Used:

`gpio_identify`

Section 4—Local Loopback from PB Interface Chip

```
Minimum Mode Required         : Destructive

Terminal Used for Execution : Any

In Default Set?               : No

In Auto-diagnostic Set?       : No
```

The local loopback tests will determine the operational status of the backplane of the GPIO Device Adapter. This will be accomplished by writing and reading data to and from (respectively) the GPIO DA via GPIODAD.

In this section, data will be written to the PB Interface Chip PB Interface Chip and read back, testing the hardware layer "just below" the backplane.

Note The PB Interface chip is used to interface the PB backplane to the midplane of the GPIO DA.

GPIODAD will compare the data read with the data written, and display the appropriate messages.

Possible Output Messages:

```
Section 4 -- Local Loopback from PB Interface Chip

   NO ERRORS DETECTED while executing loopback from the PB
   Interface Chip.

End of Section 4 -- Local Loopback from PB Interface Chip
```

Possible Error/Warning Messages:

```
*** ERROR -- PB INTERFACE CHIP (PBIC) LOOPBACK FAILED.          (GDADERR 5010)
***
***          Data written to PBIC (in hex): !
***          Data read  from PBIC (in hex): !
```

GPIO DAR Operations Used:

```
gpio_read_reg
gpio_write_reg
```

Section 5—Hardware Test

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

Executing this section will perform a thorough hardware test on the GPIO DA. Not only will the PB Interface chip be tested, but the glue logic and backplane transceivers will also be tested. The only components that will be untested after this section is executed are the glue logic that is inaccessible to `GPIODAD` . In order to run this section, a loopback hood must first be connected to the respective device adapter. After the user confirms that the loopback hood has been connected, the diagnostic performs the following specific tests:

RESET CARD - a CMD_RESET is sent to the IO_COMMAND register on the card

INTERNAL REGISTER TEST - various patterns (0, all 1s, 5s, As, walking 1) are written out to and read back from the IO_EIM, IO_DMA_LINK, IO_DMA_COUNT registers

READ IO_STATUS - verifies that the card is ready by reading bit 25 (RDY) of the IO_STATUS register

READ IODC - verifies that the 8-byte IODC on the card is readable and contains correct information for the Soft Physical Address and Module Type entries

EXTERNAL REGISTER TEST - verifies the path to/from the external registers on the card by writing out then reading back a series of bit patterns: 0s, 1s, 5s, As, and walking 1. (Information written to bits 0,1,2,3,4 of the IO_GPIO_CONTROL register is read in from bits 3,4,5,6,7 of the IO_GPIO_STATUS register (since the external loopback hood maps the corresponding pins across.)

FRONT-PLANE TRANSFER TEST - The data path from the backplane circuitry through to and including the frontplane drivers/receivers is tested by writing out and reading back a series of bit-patterns under varying configurations of the card. The patterns: all 0's, all 1's, 5's, A's, walking 1's are sent out to the IO_GPIO_OUTHI/LO registers and read in from the IO_GPIO_INHI/LO registers (which are tied to the output registers by the external loopback hood) under width={16-bit, 8-bit}, pctl/pflg={0/0,1/1}, and mode=full.

DMA TEST NUMBER 1 - This test will DMA OUT one byte of data (or two if width is 16-bit). DMA completion will be determined via the DAM (driver) receiving an interrupt from the card. The data DMA'd OUT will be verified to be correct by doing a direct register read of the frontplane input register(s) (which are tied to the output register(s) by the external loopback hood).

DMA TEST NUMBER 2	- This test will DMA OUT 500 bytes of data. DMA completion will be determined via the DAM receiving an interrupt from the card. The last byte of data (or two if width is 16-bit) DMA'd OUT will be verified to be correct by doing a direct register read of the frontplane input register(s) (which are tied to the output register(s) by the external loopback hood).
DMA TEST NUMBER 3	- This test will DMA IN one byte of data (or two if width is 16-bit). DMA completion will be determined via the DAM receiving an interrupt from the card. The data DMA'd IN will be verified to be correct by comparing the contents of the buffer it is stored into with the data that was put on the frontplane output register(s) by Direct I/O (and looped in to the frontplane input register(s) by the external loopback hood).
DMA TEST NUMBER 4	- This test will DMA IN 500 bytes of data. DMA completion will be determined via the DAM receiving an interrupt from the card. The data DMA'd IN will be verified to be correct by comparing the contents of the buffer it is stored into with the data that was put on the frontplane output register(s) by Direct I/O (and looped in to the frontplane input register(s) by the external loopback hood).
SETTLING TIME TEST	- This test performs a simple one-byte DMA OUT over all values 0 ... 15 of settling time in the IO_GPIO_DELAY register. Data is not verified, but the diagnostic checks that the EOC bit in the Shazam IO_STATUS register (bit 31) is on.
DINCLK TEST	- The ability of the card to latch data on the PFLG ready to busy transition is verified. The DINCLK source in the IO_GPIO_DELAY register is set to a value of 100 and data put out on the frontplane output registers is checked for on the frontplane input registers before and after the event causing the latch (which are tied to the output register(s) by the external loopback hood).
INTER-RUPTS TEST	- An external interrupt is generated and verified as having been recognized by the card. A 1 is written to bit CON4 of the IO_GPIO_CONTROL register to set up for the interrupt, external interrupts are enabled, and finally a 0 is written to CON4 to cause the interrupt (since the CON4 pin is tied to the interrupt line by the external loopback hood). After the interrupt, the diagnostic verifies that the NINTIN bit of Shazam's DIAGNOSTIC_STATUS register is off.
NRESET TEST	- The card is reset and the pattern AA is written to the IO_GPIO_CONTROL register. The CON0-4 bits should be correctly mapped onto STS0-4 by the loopback hood. This is verified by a read of the IO_GPIO_STATUS register. Then, the card is reset again and the STS0-4 bits are checked. They should now be all 1s.

Possible Output Messages:

```
Section 5 -- Hardware Test

   Is the loopback hood connected to the respective GPIO Device Adapter?
   (Y/N) [Y] :

   If you wish to execute this section, please connect it.

   NO ERRORS DETECTED while checking the Device Adapter for
   READY status.

   NO ERRORS DETECTED whie checking the Device Adapter's
   SPA entry in the IODC.

   NO ERRORS DETECTED while checking the Device Adapter's
   Module Type entry in the IODC.

   NO ERRORS DETECTED while checking the Device Adapter's
   External Registers: IO_GPIO_CONTROL and IO_GPIO_STATUS.

   NO ERRORS DETECTED while checking the PB Interface Chip's
   IO_EIM register.

   NO ERRORS DETECTED while checking the PB Interface Chip's
   IO_DMA_LINK register.

   NO ERRORS DETECTED while checking the PB Interface Chip's
   IO_DMA_COUNT register.

   Initiating Hardware Test Loopback ...

   Initiating Hardware Test DMA ...

   NO ERRORS DETECTED during "DMA" testing.
   Current configuration: #
   Current test number: #

   NO ERRORS DETECTED during ANY of the "DMA" Tests.

   NO ERRORS DETECTED during the "Settling Time" Test.

   NO ERRORS DETECTED during the "External Interrupts" Test.

   NO ERRORS DETECTED during the "NRESET" Test.

End of Section 5 -- Hardware Test
```

Possible Error/Warning Messages:

```
*** ERROR -- READY STATUS TEST FAILED.                         (GDADERR 5011)
***          The IO_STATUS register indicates that the Device Adapter
***          is NOT READY.

*** ERROR -- INCORRECT IODC SOFT PHYSICAL ADDRESS.             (GDADERR 5012)
***          The IODC contains an SPA entry of #.
***          This is NOT the expected value.

*** ERROR -- INCORRECT IODC MODULE TYPE                        (GDADERR 5013)
***          The IODC contains a Module Type entry of #.
***          This is NOT the expected value.

*** ERROR -- EXTERNAL REGISTER TEST FAILED.                    (GDADERR 5014)
***          Data written to IO_GPIO_CONTROL (in hex): #
***          Data read back from IO_GPIO_STATUS (in hex): #

*** ERROR -- INTERNAL REGISTER TEST FAILED.                    (GDADERR 5015)
***          Failure encountered while writing to / reading from
***          the Shazam IO_EIM register.

*** ERROR -- INTERNAL REGISTER TEST FAILED.                    (GDADERR 5016)
***          Failure encountered while writing to / reading from
***          the Shazam IO_DMA_LINK register.

*** ERROR -- INTERNAL REGISTER TEST FAILED.                    (GDADERR 5017)
***          Failure encountered while writing to / reading from
***          the Shazam IO_DMA_COUNT register.

*** ERROR -- FRONTPLANE TRANSFER TEST FAILED.                  (GDADERR 5018)
***          Timeout occurred while performing a direct loopback to
***          the Device Adapter's frontplane.
***          Current configuration: #

*** ERROR -- FRONTPLANE TRANSFER TEST FAILED.                  (GDADERR 5019)
***          Data corrupted during direct loopback to the Device Adapter's
***          frontplane.
***
***          Data written out  (in hex): #
***          Data read back in (in hex): #
***          Current configuration: #
```

```
*** ERROR -- FAILURE DURING DMA TEST.                            (GDADERR 5030)
***          The Device Adapter did not indicate READY status before
***          the DMA transfer began.
***          Current configuration: #
***          Current test number: #

*** ERROR -- FAILURE DURING DMA TEST.                            (GDADERR 5031)
***          Incorrect residue value in the IO_DMA_COUNT register.

*** ERROR -- FAILURE DURING DMA TEST.                            (GDADERR 5032)
***          A soft error was indicated by the IO_STATUS register after DMA.
***          Current configuration: #
***          Current test number: #

*** ERROR -- FAILURE DURING DMA TEST.                            (GDADERR 5033)
***          A fatal error was indicated by the IO_STATUS register after DMA.
***          Current configuration: #
***          Current test number: #

*** ERROR -- FAILURE DURING DMA TEST.                            (GDADERR 5034)
***          The IO_STATUS register indicates NOT READY after DMA.
***          Current configuration: #
***          Current test number: #

*** ERROR -- FAILURE DURING DMA TEST.                            (GDADERR 5035)
***          The EOC bit in the IO_STATUS register is off after DMA.
***          Current configuration: #
***          Current test number: #

*** ERROR -- FAILURE DURING SMALL DMA OUT TEST.                  (GDADERR 5036)
***          The data dma transferred out was corrupted.
***          Current configuration: #
***          Current test number: #
***          Data intended to be sent out (in hex): #
***          Data actually read in to verify (in hex): #

*** ERROR -- FAILURE DURING LARGE DMA OUT TEST.                  (GDADERR 5037)
***          The data dma transferred out was corrupted.
***          Current configuration: #
***          Current test number: #
***          Data intended to be sent out (in hex): #
***          Data actually read in to verify (in hex): #
```

```
*** ERROR -- FAILURE DURING SMALL DMA IN TEST.                 (GDADERR 5038)
***          The data dma transferred in was corrupted.
***          Current configuration: #
***          Current test number: #
***          Data directly sent out (in hex): #
***          Data dma'd in (in hex): #

*** ERROR -- FAILURE DURING LARGE DMA IN TEST.                 (GDADERR 5039)
***          The data dma transferred in was corrupted.
***          Current configuration: #
***          Current test number: #
***          Data directly sent out (in hex): #
***          Data dma'd in (in hex): #

*** ERROR -- FAILURE DURING SETTLING TIME TEST.                (GDADERR 5040)
***          The Device Adapter was NOT READY before DMA transfer.

*** ERROR -- FAILURE DURING SETTLING TIME TEST.                (GDADERR 5041)
***          The EOC bit in the IO_STATUS register is off after DMA.

*** ERROR -- FAILURE DURING DINCLK TEST.                       (GDADERR 5043)
***          Normal data transfer could not be validated before varying
***          the DINCLK source to #.

*** ERROR -- FAILURE DURING DINCLK TEST.                       (GDADERR 5044)
***          Data was not correctly latched with the DINCLK source
***          set to #.

*** ERROR    FAILURE DURING EXTERNAL INTERRUPTS TEST.          (GDADERR 5045)
***          After asserting the interrupt line, the IO_STATUS register
***          did not indicate the correct external interrupt status.

*** ERROR    FAILURE DURING NRESET TEST.                       (GDADERR 5046)
***          The IO_STATUS register did not contain the correct contents
***          during a reset test.
```

GPIO DAR Functions Used:

```
gpio_reset_card
gpio_read_reg
gpio_write_reg
gpio_input
gpio_output
```

Section 6—Status

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : Yes

In Auto-diagnostic Set?      : No
```

This section is broken down into four steps so that the diagnostician can convey to GPIODAD precisely what status information he/she wishes to examine. Moreover, the diagnostician can have a very thorough status report of the GPIO Device Adapter if he/she desires it by executing all four steps. In order to make this ample supply of status information easier for the diagnostician to assimilate, it is fully decoded and transformed into user-friendly messages by GPIODAD before being displayed to the diagnostic user.

If this section is run without the desired steps being explicitly specified, Step 61 will run by default.

Step 61—Read PB Interface Chip IO_STATUS Register

Executing this step will display to the user various module specific status information in a user-friendly manner.

Possible Output Messages:

```
Section 6 -- Status

   Step 61 - Read PB Interface Chip IO_STATUS Register

         An interrupt has NOT occurred since the last write to
         IO_COMMAND.                                          (Bit  0 = 0)

         An interrupt message HAS been sent since the last write
         to the IO_COMMAND register.                          (Bit  0 = 1)

         An interrupt message was NOT issued due to circuitry external
         to the PB Interface Chip.                            (Bit  2 = 0)

         An interrupt HAS been issued due to circuitry external to
         the PB Interface Chip.                               (Bit  2 = 1)

         Transfer NOT completed.                              (Bit  3 = 0)

         Transfer completed.                                  (Bit  3 = 1)

         A soft error (length conflict) has NOT occurred since the
         last write to the IO_COMMAND register.               (Bit 22 = 0)

         A fatal error has NOT occurred since the last write to
         the IO_COMMAND register.                             (Bit 24 = 0)

         The device adapter is NOT ready for a new command.   (Bit 25 = 0)

         The device adapter IS ready for a new command.       (Bit 25 = 1)

         An interrupt message has NOT been sent since last
         ii_clear.                                            (Bit 26 = 0)

         An interrupt message HAS been sent since last
         ii_clear.                                            (Bit 26 = 1)

         The EOC bit of the IO_DMA_LINK register
         is NOT set.                                          (Bit 31 = 0)

         The EOC bit of the IO_DMA_LINK register IS set.      (Bit 31 = 1)

      A binary representation of the respective register(s) follows:
      NOTE: Dashes represent undefined bits.
```

```
----------------------------------------------------------
IO_STATUS register's image:

Bit #:     0     4     8     12    16    20    24    28

Value:     #-##  ----  ----  ----  ----  --#-  ###-  ---#
----------------------------------------------------------
```

End of Step 61 - Read PB Interface Chip IO_STATUS Register

End of Section 6 -- Status

Possible Error/Warning Messages:

```
*** WARNING -- The IO_STATUS register has one or more undefined bits
***            reading in as a 1 (I expected all 0's).         (GDADWARN 6034)

*** WARNING -- A SOFT ERROR (length conflict) occurred since the last
***            write to the IO_COMMAND register.
***            (IO_STATUS Bit 22 = 1)                          (GDADWARN 6031)

*** WARNING -- A FATAL ERROR occurred since the last write to the
***            IO_COMMAND register.
***            (IO_STATUS Bit 24 = 1)                          (GDADWARN 6032)
```

GPIO DAR Operations Used:

gpio_read_reg

Step 62—Read PB Interface Chip DIAGNOSTIC_STATUS Register

When this step is executed, the respective register will be read in, decoded and displayed in a user-friendly fashion. The information output deals mainly with the internal state of the PB Interface Chip.

Possible Output Messages:

```
Section 6 -- Status

   Step 62 -- Read PB Interface Chip DIAGNOSTIC_STATUS Register

      PB Interface Chip DIAGNOSTIC_STATUS Register Data:

         Circuitry external to the PB Interface Chip HAS
           issued an interrupt request.                    (Bit  0 = 0)

         Circuitry external to the PB Interface Chip HAS NOT
           issued an interrupt request.                    (Bit  0 = 1)

         The PB Interface Chip did NOT generate a PB error
           since the last reset command.                   (Bit  1 = 1)

         DMA is DISABLED.                                  (Bit  4 = 0)
         DMA is ENABLED.                                   (Bit  4 = 1)

         Interrupt message transmission is DISABLED.       (Bit 19 = 0)
         Interrupt message transmission is ENABLED.        (Bit 19 = 1)

         The FIFO is NOT FULL.                             (Bit 20 = 0)
         The FIFO is FULL.                                 (Bit 20 = 1)

         The FIFO is NOT EMPTY.                            (Bit 21 = 0)
         The FIFO is EMPTY.                                (Bit 21 = 1)

         There is 1 byte in the FIFO.                      (Bits 23..27)
         There are ! bytes in the FIFO.                    (Bits 23..27)

      A binary representation of the respective register(s) follows:
      NOTE: Dashes represent undefined bits.

         --------------------------------------------------------
         DIAGNOSTIC_STATUS register's image:

         Bit #:     0     4     8     12    16    20    24    28

         Value:     ####  ####  ####  ####  ####  ####  ####  ####
         --------------------------------------------------------

   End of Step 62 - Read PB Interface Chip DIAGNOSTIC_STATUS Register
```

```
End of Section 6 -- Status
```

Possible Error/Warning Messages:

```
*** WARNING -- PB Interface Chip HAS generated a PB bus error since
***             the last reset command.
***             (DIAGNOSTIC_STATUS Bit 1 = 0)                     (GDADWARN 6040)

*** ERROR -- PB INTERFACE CHIP FAILURE.                          (GDADERR 5071)
***
***          PB Interface Chip reported to have ! bytes in FIFO,
***          however only 24 bytes are available.

*** ERROR -- PB INTERFACE CHIP FAILURE.                          (GDADERR 5072)
***
***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) is set
***          (implying FIFO CNT should = 24); however, the FIFO CNT field
***          (Bits 23..27) = ! (decimal).

*** ERROR -- PB INTERFACE CHIP FAILURE.                          (GDADERR 5073)
***
***          DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) is set
***          (implying FIFO CNT should = 0); however, the FIFO CNT field
***          (Bits 23..27) = ! (decimal).

*** ERROR -- PB INTERFACE CHIP FAILURE.                          (GDADERR 5074)
***
***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) and
***          FIFO EMPTY bit (Bit 21) are both set.

*** ERROR -- PB INTERFACE CHIP FAILURE.                          (GDADERR 5076)
***
***          DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) = 0
***          (implying FIFO CNT should be less than 24); however, the
***          FIFO CNT field (Bits 23..27) = 24.

*** ERROR -- PB INTERFACE CHIP FAILURE.                          (GDADERR 5075)
***
***          DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) = 0
***          (implying FIFO CNT should be greater than 0); however, the
***          FIFO CNT field (Bits 23..27) = 0.
```

GPIO DAR Operations Used:

gpio_read_reg

Step 63—Read Device Adapter IO_GPIO_STATUS Register

Running this step will result in status information of the interface with the peripheral to be displayed in a user-friendly fashion.

Possible Output Messages:

```
Section 6 -- Status

   Step 63 - Read Device Adapter IO_GPIO_STATUS Register

      The logic levels of the respective bits are as follows:

                   Line ID                            Logic Level
                   -------                            -----------
         RDY - Frontplane State           (Bit 0):         #
               (Frontplane state READY.)
               (Frontplane state BUSY.)

        PFLG - Peripheral Flag            (Bit 1):         #

        PCTL - Peripheral Control         (Bit 2):         #

        EINT/STS4 - EINT/X-Status Line 4 (Bit 3):          #

        STS0 - Extended Status Line 0     (Bit 4):         #
        STS1 - Extended Status Line 1     (Bit 5):         #
        STS2 - Extended Status Line 2     (Bit 6):         #
        STS3 - Extended Status Line 3     (Bit 7):         #


      A binary representation of the respective register(s) follows:
      NOTE: Dashes represent undefined bits.

        --------------------------------------------------------
        IO_GPIO_STATUS register's image:

        Bit #:     0     4     8    12    16    20    24    28

        Value:    ----  ----  ----  ----  ----  ----  ####  ####
        --------------------------------------------------------

   End of Step 63 - Read Device Adapter IO_GPIO_STATUS Register

End of Section 6 -- Status
```

Possible Error/Warning Messages:

None specified by the diagnostic.

GPIO DAR Operations Used:

gpio_read_reg

Step 64—Read Device Adapter IO_GPIO-INHI/LO Registers

This step is executed when the user desires to view the data within the IO_GPIO_INHI and IO_GPIO_INLO registers.

Possible Output Messages:

```
Section 6 -- Status

   Step 64 - Read Device Adapter IO_GPIO_INHI/LO Registers

      A binary representation of the respective register(s) follows:
      NOTE: Dashes represent undefined bits.

         ----------------------------------------------------------
         IO_GPIO_INHI register's image:

         Bit #:     0     4     8     12    16    20    24    28

         Value:     ----  ----  ----  ----  ----  ----  ####  ####
         ----------------------------------------------------------
         IO_GPIO_INLO register's image:

         Bit #:     0     4     8     12    16    20    24    28

         Value:     ----  ----  ----  ----  ----  ----  ####  ####
         ----------------------------------------------------------

   End of Step 64 - Read Device Adapter IO_GPIO_INHI/LO Registers

End of Section 6 -- Status
```

Possible Error/Warning Messages:

None specified by the diagnostic.

GPIO DAR Operations Used:

gpio_read_reg

Section 8—External Loopback

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

Executing this section allows the user to test the data path of the entire device adapter, including the frontplane drivers/receivers. However, in order to run this section, a loopback hood must first be connected to the respective device adapter. After the user confirms that the loopback hood has been connected, the diagnostic performs a loopback test via the frontplane (since the output and input registers are tied together by the hood) and the results are displayed to the user.

Note A loopback hood must be connected to the respective GPIO Device Adapter before this section is executed, or else a timeout error will be reported and the diagnostic will terminate.

Possible Output Messages:

```
Section 8 -- External Loopback

   Is the loopback hood connected to the respective GPIO Device Adapter?
   (Y/N) [Y] :

   Initiating Loopback ...

   If you wish to execute this section, please connect it.

   NO ERRORS DETECTED while executing the external loopback via the GPIO
   Device Adapter's frontplane.

End of Section 8 -- External Loopback
```

Possible Error/Warning Messages:

```
*** ERROR -- EXTERNAL LOOPBACK FAILED.                          (GDADERR 5080)
***
***         Data written to frontplane (in hex): !
***         Data read  from frontplane (in hex): !

*** ERROR -- EXTERNAL LOOPBACK TIMED OUT.                       (GDADERR 5082)
***
***         The IO_GPIO_STATUS register did not indicate ready
***         after waiting an appropriate delay for the transfer
***         to complete.
```

GPIO DAR Operations Used:

```
gpio_read_reg
gpio_write_reg
```

Section 10—Register Level Input/Output Transactions

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

Warning **The user of this section must exercise EXTREME CAUTION when sending information over the GPIO.**

This section can be executed by the diagnostician when he/she wishes to do "peeks" and/or "pokes" to the device adapter. That is to say, this section allows the user to read/write **any** register on the device adapter's address space.

Warning **In order for this section to be of any value, the device adapter being diagnosed should not be accessed by any process other than `GPIODAD`. This is to ensure that any configuration modifications made by the diagnostic are pending for succeeding transactions.**

Possible Output Messages:

```
Section 10 -- Register Level Input/Output Transactions

   At the prompt, enter one of the following commands:
     1) "Input Register" Command:

             i <decimal register number>

        For Example:
          To input register number 159, type "i 159<CR>"

     2) "Output Register" Command:

             o <decimal register number> <hex data>

        For Example:
          To output 0x02BAD to register number 12 (decimal), type
          "o 12 2bad<CR>"

     3) A lone <CR> will cause this section to terminate.

   >>

    : !

End of Section 10 -- Register Level Input/Output Transactions
```

Note All commands/data can be entered in upper and/or lower case characters.

Possible Error/Warning Messages:

```
*** WARNING -- This section is intended for personnel which have a good
***            understanding of the GPIO Device Adapter's architecture.
***            It is strongly recommended that you print off "More Help" on
***            Section 10, or reference the user's manual before using this
***            section.                                      (GDADWARN 6050)

*** WARNING -- Invalid or missing register number ... the register number
***            value is not a properly formatted DECIMAL integer.
***            Please try again.                             (GDADWARN 6051)

*** WARNING -- Invalid register number ... the register number entered
***            is beyond the device adapter's address space (highest
***            register number is decimal 1023).
***            Please try again.                             (GDADWARN 6052)

*** WARNING -- Invalid or missing datum ... the datum value is not a properly
***            formatted HEXADECIMAL integer.
***            Please try again.                             (GDADWARN 6053)

*** WARNING -- Invalid command ... the command entered cannot be decoded.
***            Please try again.                             (GDADWARN 6054)
```

GPIO DAR Operations Used:

gpio_write_reg
gpio_read_reg

Section 11—Data Transaction on GPIO

```
Minimum Mode Required        : Destructive

Terminal Used for Execution : Any

In Default Set?              : No

In Auto-diagnostic Set?      : No
```

Executing this section will allow the user to configure the GPIO card and then perform a data transaction with a device connected to the interface. If the user selects 8-bit operation, a single byte of data can be sent or received over the GPIO. If the user selects 16-bit operation, two bytes of data can be sent (the high byte is entered by the user and the low byte is all 0's) or received over the GPIO. This implies that the user must have a very good working knowledge of the particular device protocol.

When this section is run, single-byte data entered is expected to be in hexadecimal format. Also, the user may restart information entry by entering a lone semicolon (;) at any prompt. Entering a lone (:) will convey to the diagnostic that a byte (or two) of return data is expected from a device on the GPIO. When expected data is returned within the timeout period, it will be displayed in hexadecimal format; however, if the timeout period elapses, an appropriate error message will be output.

Possible Output Messages:

```
Section 11 -- Data Transaction on GPIO


      WIDTH - Bit 0 of IO_GPIO_CONFIG Register
        Do you wish to have 8-bit (enter 0) or 16-bit (enter 1) DMA
        transfers on the frontplane?
        (0/1) [0] :

      PFLG - Bit 1 of IO_GPIO_CONFIG Register
        Do you wish to have PFLG indicate "Ready" when it is at a logic
        low state (enter 0) or when it is at a logic high state (enter 1)?
        (0/1) [0] :

      PCTL - Bit 2 of IO_GPIO_CONFIG Register
        Do you wish to have PCTL set control when it is at a logic low
        state (enter 0) or when it is at a logic high state (enter 1)?
        (0/1) [0] :

      PDDR - Bit 3 of IO_GPIO_CONFIG Register
        Do you wish to have PDDR signify the "out" direction when it is
        at a logic low state (enter 0) or when it is at a logic high
        state (enter 1)?
        (0/1) [0] :

      MODE - Bit 7 of IO_GPIO_CONFIG Register
        Handshake Modes:  0 = Full Mode (or going to set Strobed)
```

```
                      1 = Pulsed Mode
    Enter the number corresponding to the desired handshake mode.
    (0/1) [0] :

  STROBE - Bit 6 of IO_GPIO_CONFIG Register
    Handshake Modes:  0 = other mode already set
                      1 = Strobed Mode
    Enter the number corresponding to the desired handshake mode.
    (0/1) [0] :

 At a prompt, enter one of the following:
 1) A hexadecimal integer (8-bit maximum -- i.e. 0..FF)
 2) A lone colon (:) to convey to the diagnostic that a byte of data
    from the attached device is expected
 3) A lone semicolon (;) to abort the current information entry session
    and to start over

CAUTION: Care must be taken not to enter a hexadecimal integer that
matches a language localized control message.  For example, if a lone
'e' is entered, it is an indication to the diagnostic to exit.  In this
case, "0e" should be entered.

>>

The device RETURNED the following data (low byte only if 8-bit width) -
HIGH BYTE (in hex format):
LOW  BYTE (in hex format):

... Information entry to be restarted.

Information being sent to GPIO output register ...

End of Section 11 -- Data Transaction on GPIO
```

Possible Error/Warning Messages:

```
*** WARNING -- It is strongly recommended that you print off "More Help" on
***            Section 11, or reference the user's manual before using this
***            section.                                    (GDADWARN 6070)

*** WARNING -- Information entered is not a properly formatted
***            HEXADECIMAL integer.                        (GDADWARN 6075)
***
***            Try it again.  For example: "2E<CR>".

*** WARNING -- Information entered exceeds the 8-bit limit.    (GDADWARN 6076)
***
***            Try it again.  For example: "FF<CR>".
```

GPIO DAR Operations Used:

```
gpio_write_reg
gpio_read_reg
gpio_reset_card
```

Error and Warning Messages

The following is a list of the most significant error and warning messages displayed by the system, arranged in numerical order.

Error Messages

The following is a list of the most significant error messages displayed by the system, arranged in numerical order.

5010

```
*** ERROR -- PB INTERFACE CHIP (PBIC) LOOPBACK FAILED.          (GDADERR
5010)
***
***         Data written to PBIC (in hex): !
***         Data read  from PBIC (in hex): !
```

5011

```
*** ERROR -- READY STATUS TEST FAILED.                          (GDADERR
5011)
***         The IO_STATUS register indicates that the Device Adapter
***         is NOT READY.
```

5012

```
*** ERROR -- INCORRECT IODC SOFT PHYSICAL ADDRESS.              (GDADERR
5012)
***         The IODC contains an SPA entry of #.
***         This is NOT the expected value.
```

5013

```
*** ERROR -- INCORRECT IODC MODULE TYPE                         (GDADERR
5013)
***         The IODC contains a Module Type entry of #.
***         This is NOT the expected value.
```

5014

```
*** ERROR -- EXTERNAL REGISTER TEST FAILED.                     (GDADERR
5014)
***         Data written to IO_GPIO_CONTROL (in hex): #
***         Data read back from IO_GPIO_STATUS (in hex): #
```

5015 *** ERROR -- INTERNAL REGISTER TEST FAILED. (GDADERR 5015)

*** Failure encountered while writing to / reading from

*** the Shazam IO_EIM register.

5016 *** ERROR -- INTERNAL REGISTER TEST FAILED. (GDADERR 5016)

*** Failure encountered while writing to / reading from

*** the Shazam IO_DMA_LINK register.

5017 *** ERROR -- INTERNAL REGISTER TEST FAILED. (GDADERR 5017)

*** Failure encountered while writing to / reading from

*** the Shazam IO_DMA_COUNT register.

5018 *** ERROR -- FRONTPLANE TRANSFER TEST FAILED. (GDADERR 5018)

*** Timeout occurred while performing a direct loopback to

*** the Device Adapter's frontplane.

*** Current configuration: #

5019 *** ERROR -- FRONTPLANE TRANSFER TEST FAILED. (GDADERR 5019)

*** Data corrupted during direct loopback to the Device Adapter's

*** frontplane.

***

*** Data written out (in hex): #

*** Data read back in (in hex): #

*** Current configuration: #

5030 *** ERROR -- FAILURE DURING DMA TEST. (GDADERR 5030)

*** The Device Adapter did not indicate READY status before

*** the DMA transfer began.

*** Current configuration: #

*** Current test number: #

5031 *** ERROR -- FAILURE DURING DMA TEST. (GDADERR 5031)

*** Incorrect residue value in the IO_DMA_COUNT register.

5032 *** ERROR -- FAILURE DURING DMA TEST. (GDADERR 5032)

*** A soft error was indicated by the IO_STATUS register after DMA.

*** Current configuration: #

*** Current test number: #

5033 *** ERROR -- FAILURE DURING DMA TEST. (GDADERR 5033)

*** A fatal error was indicated by the IO_STATUS register after DMA.

*** Current configuration: #

*** Current test number: #

5034 *** ERROR -- FAILURE DURING DMA TEST. (GDADERR 5034)

*** The IO_STATUS register indicates NOT READY after DMA.

*** Current configuration: #

*** Current test number: #

5035 *** ERROR -- FAILURE DURING DMA TEST. (GDADERR 5035)

*** The EOC bit in the IO_STATUS register is off after DMA.

*** Current configuration: #

*** Current test number: #

5036 *** ERROR -- FAILURE DURING SMALL DMA OUT TEST. (GDADERR 5036)

*** The data dma transferred out was corrupted.

*** Current configuration: #

*** Current test number: #

*** Data intended to be sent out (in hex): #

*** Data actually read in to verify (in hex): #

5037 *** ERROR -- FAILURE DURING LARGE DMA OUT TEST. (GDADERR 5037)

*** The data dma transferred out was corrupted.

*** Current configuration: #

*** Current test number: #

*** Data intended to be sent out (in hex): #

*** Data actually read in to verify (in hex): #

5038 *** ERROR -- FAILURE DURING SMALL DMA IN TEST. (GDADERR 5038)

*** The data dma transferred in was corrupted.

*** Current configuration: #

*** Current test number: #

*** Data directly sent out (in hex): #

*** Data dma'd in (in hex): #

5039 *** ERROR -- FAILURE DURING LARGE DMA IN TEST. (GDADERR 5039)

*** The data dma transferred in was corrupted.

*** Current configuration: #

*** Current test number: #

*** Data directly sent out (in hex): #

*** Data dma'd in (in hex): #

5040 *** ERROR -- FAILURE DURING SETTLING TIME TEST. (GDADERR 5040)

*** The Device Adapter was NOT READY before DMA transfer.

5041 *** ERROR -- FAILURE DURING SETTLING TIME TEST. (GDADERR 5041)

*** The EOC bit in the IO_STATUS register is off after DMA.

5043 *** ERROR -- FAILURE DURING DINCLK TEST. (GDADERR 5043)

*** Normal data transfer could not be validated before varying

*** the DINCLK source to #.

5044 *** ERROR -- FAILURE DURING DINCLK TEST. (GDADERR 5044)

*** Data was not correctly latched with the DINCLK source

*** set to #.

5045 *** ERROR FAILURE DURING EXTERNAL INTERRUPTS TEST. (GDADERR 5045)

*** After asserting the interrupt line, the IO_STATUS register

*** did not indicate the correct external interrupt status.

5046 *** ERROR FAILURE DURING NRESET TEST. (GDADERR 5046)
*** The IO_STATUS register did not contain the correct contents
*** during a reset test.

5071 *** ERROR -- PB INTERFACE CHIP FAILURE. (GDADERR 5071)
***
*** PB Interface Chip reported to have ! bytes in FIFO,
*** however only 24 bytes are available.

5072 *** ERROR -- PB INTERFACE CHIP FAILURE. (GDADERR 5072)
***
*** DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) is set
*** (implying FIFO CNT should = 24); however, the FIFO CNT field
*** (Bits 23..27) = ! (decimal).

5073 *** ERROR -- PB INTERFACE CHIP FAILURE. (GDADERR 5073)
***
*** DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) is set
*** (implying FIFO CNT should = 0); however, the FIFO CNT field
*** (Bits 23..27) = ! (decimal).

5074 *** ERROR -- PB INTERFACE CHIP FAILURE. (GDADERR 5074)
***
*** DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) and
*** FIFO EMPTY bit (Bit 21) are both set.

```
5075    *** ERROR -- PB INTERFACE CHIP FAILURE.                    (GDADERR
        5075)
        ***
        ***         DIAGNOSTIC_STATUS register's FIFO EMPTY bit (Bit 21) = 0
        ***         (implying FIFO CNT should be greater than 0); however, the
        ***         FIFO CNT field (Bits 23..27) = 0.
```

```
5076    *** ERROR -- PB INTERFACE CHIP FAILURE.                    (GDADERR
        5076)
        ***
        ***         DIAGNOSTIC_STATUS register's FIFO FULL bit (Bit 20) = 0
        ***         (implying FIFO CNT should be less than 24); however, the
        ***         FIFO CNT field (Bits 23..27) = 24.
```

```
5080    *** ERROR -- EXTERNAL LOOPBACK FAILED.                     (GDADERR
        5080)
        ***
        ***         Data written to frontplane (in hex): !
        ***         Data read  from frontplane (in hex): !
```

```
5082    *** ERROR -- EXTERNAL LOOPBACK TIMED OUT.                  (GDADERR
        5082)
        ***
        ***         The IO_GPIO_STATUS register did not indicate ready
        ***         after waiting an appropriate delay for the transfer
        ***         to complete.
```

Warning Messages

The following is a list of the most significant warning messages displayed by the system, arranged in numerical order.

6001

```
*** WARNING -- Invalid response.  Please answer the question with one
***            of the choices given.
(GDADWARN 6001)
More Help (1..6, 8, 10, 11, <CR>) :
```

6010

```
*** WARNING -- Soft Physical Address Capability = !,
***            I expected a 0.
(GDADWARN 6010)
```

6011

```
*** WARNING -- Type of Module = ! (UNKNOWN PRODUCT),
***            I expected a 4 (TP_A_DMA).
(GDADWARN 6011)
```

6031

```
*** WARNING -- A SOFT ERROR (length conflict) occurred since the last
***            write to the IO_COMMAND register.
***            (IO_STATUS Bit 22 = 1)
(GDADWARN 6031)
```

6032

```
*** WARNING -- A FATAL ERROR occurred since the last write to the
***            IO_COMMAND register.
***            (IO_STATUS Bit 24 = 1)
(GDADWARN 6032)
```

6034 `*** WARNING -- The IO_STATUS register has one or more undefined bits`
```
***            reading in as a 1 (I expected all 0's).
(GDADWARN 6034)
```

6040
```
*** WARNING -- PB Interface Chip HAS generated a PB bus error since
***            the last reset command.
***            (DIAGNOSTIC_STATUS Bit 1 = 0)
(GDADWARN 6040)
```

6050
```
*** WARNING -- This section is intended for personnel which have a good
***            understanding of the GPIO Device Adapter's architecture.
***            It is strongly recommended that you print off "More Help"
on
***            Section 10, or reference the user's manual before using
this
***            section.
(GDADWARN 6050)
```

6051
```
*** WARNING -- Invalid or missing register number ... the register
number
***            value is not a properly formatted DECIMAL integer.
***            Please try again.
(GDADWARN 6051)
```

6052
```
*** WARNING -- Invalid register number ... the register number entered
***            is beyond the device adapter's address space (highest
***            register number is decimal 1023).
***            Please try again.
(GDADWARN 6052)
```

6053 *** WARNING -- Invalid or missing datum ... the datum value is not a properly

*** formatted HEXADECIMAL integer.

*** Please try again.
(GDADWARN 6053)

6054 *** WARNING -- Invalid command ... the command entered cannot be decoded.

*** Please try again.
(GDADWARN 6054)

6070 *** WARNING -- It is strongly recommended that you print off "More Help" on

*** Section 11, or reference the user's manual before using this

*** section.
(GDADWARN 6070)

6075 *** WARNING -- Information entered is not a properly formatted

*** HEXADECIMAL integer.
(GDADWARN 6075)

***

*** Try it again. For example: "2E<CR>".

6076 *** WARNING -- Information entered exceeds the 8-bit limit.
(GDADWARN 6076)

***

*** Try it again. For example: "FF<CR>".

7

CIO Asynchronous Port Multiplexer Diagnostic

Introduction

MUXDIAG is the CIO Asynchronous Port Multiplexer (MUX card) diagnostic. The diagnostic tests the card's functionality by testing communication from the SPU to the MUX card. MUXDIAG can also initiate the onboard selftests that are resident in the MUX card's EPROM. The MUX card is a Field Replaceable Unit (FRU).

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10010.

Minimum Configuration

The hardware required to run the diagnostic consists of an HP Precision Architecture computer system up and running on either the HP-UX or MPE/iX operating system.

When running under the HP-UX operating system, the following hardware must be present:

- At least one MUX card is needed. Where more than one MUX card is used, the diagnostic should be run from a terminal attached to a card not being tested.
- The system console (or a remote X-Window) is usually used to run the diagnostic. Of course, if the MUX card for the system console appears to be malfunctioning, use a terminal connected to another MUX card.

When running under the MPE/iX operating system, the following hardware is recommended:

- One MUX card
- A configured and functional LAN system
- A configured and functional Distributed Terminal Control system (DTC)

MUXDIAG can also be run from a terminal attached to the MUX card being tested. This is useful when running non-destructive test sections, such as State and Identify, or when the user does not have access to a second MUX card. When running MUXDIAG on only one MUX card, input to and output from MUXDIAG should be redirected. An example is shown in the "Examples" Section of this chapter.

Operating Instructions

MUXDIAG is accessible by all users who have obtained a diagnostic security level of 0 or 1. MUXDIAG is only executed in Single User Mode (SUM). The DUI MODE SUM command can be used to change the operating mode of the diagnostic subsystem to Single User Mode. In addition, MUXDIAG is a destructive diagnostic and cannot guarantee I/O data integrity of any processes running prior to the MUXDIAG testing. Therefore, all users on the MUX card being tested should complete their terminal I/O processes before entering Single User Mode and before running the diagnostic.

Although the users do not have to be logged off, the devices attached to the MUX card will not be accessible while MUXDIAG is running. Double check and warn the users before running the diagnostic. Refer to the DUI section for a detailed description of user capabilities, diagnostic security levels, and destructive mode. Control will return to the DUI upon completion of the default/specified sections and steps.

Default Tests

Default sections and steps are executed automatically if the user does not specify any RUN Command Options or does not specify the sections and steps to be run. The default sections and steps are:

Section 1 — **State:** Obtains and displays the current state of the MUX, which may be loaded or broken.

- Step 10: **State loaded check test**
- Step 11: **State broken check test**

Section 2 — **Clear:** Clears the MUX card and places it in a ready state. This is done by executing the internal self-test and by downloading the RAM code.

Section 3 — **Identify:** Displays the following fields of identification information: card ID, firmware ID, firmware revision, hardware revision, number of active ports, multiplexing support code, parity support, and data transfer modes.

Section 4 — **Loopback:** Performs loopback tests.

- Step 40: **Backplane loopback test**

Section 6 — **Write/Read:** Writes data to the MUX RAM, reads data back, and compares.

- Step 61: **RAM read test**

Run Command

To run the Online Diagnostics, enter the following command at the system prompt:

```
SYSDIAG
```

The system responds with the following prompt indicating that access has been gained to the Online Diagnostic Subsystem:

```
DUI >
```

Typing HELP causes a summary of the DUI and its commands to be displayed. Refer to the DUI Section of this manual for details. Enter the following commands at the DUI prompts:

```
DUI > MODE SUM                {Go into Single User Mode}
Single User Mode (SUM)        {Displayed by SYSDIAG}
```

Note

The device to be tested must be powered up and on line. Device physical locations (pdev) shown in the RUN commands are those of the devices on the "typical A1002A" system configuration described in the chapter on DUI. The pdev value entered must be correct for the system being tested.

For example, to run the diagnostic, you might enter:

```
DUI > RUN MUXDIAG pdev=4.3 <RUN Command Options>
                         |          |
                         |     none required for
                         |     default test suite
                         |
         insert physical location of
         device adapter to be tested here;
         alternatively, for MPE/iX,
         type the ldev number;
         for HP-UX, type the devfile name
```

When running MUXDIAG, it is a good idea to specify an output file to which all diagnostic messages will be sent. The messages display error codes, revision numbers, and data being written or read. Such information is helpful to the HP Customer Engineer in determining whether or not to replace the MUX card.

Test Execution

The diagnostic displays a header and welcome message.

After displaying the header, MUXDIAG calls an internal program service routine to lock the card.

The sections and steps specified in the MUXDIAG run string will be executed. The test sections and steps are described in the "Test Section Descriptions" area of this chapter. If no sections or steps were specified, the default sections will be run by MUXDIAG.

Control returns to the Online Diagnostic Subsystem upon completing the requested or default sections and steps. The following prompt is then displayed:

```
DUI >
```

If the ERRCOUNT option of the RUN command is specified by the user and the number of errors generated by MUXDIAG has reached that limit, the following message is output:

```
Error count exceeded. count = X
```

where "X" = error count originally specified in the RUN command. The diagnostic then terminates execution and returns control to the Online Diagnostic Subsystem.

If the ERRPAUSE option of the RUN command is specified by the user, the diagnostic stops after each error is generated and ask the user whether the test should continue:

```
Error pause -- do you wish to continue (y/n) [ y ]?
```

If the response is "y" or (Return), then the test resumes. If the response is "n", the diagnostic terminates.

If the LOOPCOUNT option of the RUN command is specified by the user, MUXDIAG executes a specified number of times. The following message is output before each iteration of the loop.

```
Loop Count = n
```

The value n is the iteration number starting from 1. It is incremented each time through the loop until the LOOPCOUNT has been reached. The above message is not printed if LOOPCOUNT is 1.

Test Section Descriptions

There are six diagnostic program sections available for user selection:

Section 1	**State:** Obtains and displays the current state of the MUX, which may be loaded or broken.
	Step 10: **State loaded check test**
	Step 11: **State broken check test**
Section 2	**Clear:** Clears the MUX card and places it in a ready state. This is done by executing the internal self-test and by downloading the RAM code.
Section 3	**Identify:** Displays the following fields of identification information: card ID, firmware ID, firmware revision, hardware revision, number of active ports, multiplexing support code, parity support, and data transfer modes.
Section 4	**Loopback:** Performs loopback tests.
	Step 40: **Backplane loopback test**
	Step 41: **Frontplane loopback test**
	Step 42: **Cable loopback test**
	Step 43: **Panel loopback test**
Section 5	**Selftest:** Executes the selftest program stored in the EPROM chip on the MUX card.
Section 6	**Write/Read:** Writes data to the MUX RAM, reads data back, and compares.
	Step 60: **RAM write test**
	Step 61: **RAM read test**
	Step 62: **Compare RAM write and read test**
Section 9	**Dump:** Copies the MUX RAM memory contents into a file.

Section 1 - STATE

This section displays the current state of the MUX card. Section 1 is one of the default sections that is executed if no parameters are specified in `RUN MUXDIAG`.

SAMPLE OUTPUT MESSAGE

```
Beginning Section 1
Beginning Step 10
   The MUX card RAM code is LOADED.
End of Step 10
Beginning Step 11
   The MUX card is marked as NOT BROKEN.
End of Step 11
End of Section 1
```

When the RAM code is LOADED, it has been downloaded with the proper information from the operating system. When the RAM code is NOT LOADED, the information may be incorrect and invalid. The integrity of the MUX card and testing the card will not be reliable. In this case, `MUXDIAG` displays the following message:

```
The MUX card RAM code is NOT LOADED.
```

To load the RAM, run Section 2 - Clear or reboot the operating system.

When the card is marked BROKEN the card is not functional. Either there is a hardware failure or the card has been intentionally set BROKEN to prevent its use. The following message is displayed:

```
The MUX card is marked as BROKEN.
```

To further test the MUX card run Section 4 - Loopback, Section 5 - Selftest, and/or Section 6 - Write/Read.

If any of these tests fails, replace the MUX card for it is truly malfunctioning. Refer to the hardware reference manual corresponding to your MUX for instructions on removing and replacing the MUX card.

Section 2 - CLEAR

This section puts the MUX card into a ready state by executing the card's internal selftest and by downloading the RAM code. CLEAR is destructive, and all users must complete their terminal I/O before running the test.

SAMPLE OUTPUT MESSAGE

```
Beginning Section 2

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

   The MUX card CLEARED successfully.
End of Section 2
```

It is recommended to run the Clear test before running the Section 4 - Loopback test. The Frontplane Loopback test requires that the MUX card be downloaded first, and the Clear test downloads the RAM code to the MUX card. Refer to Section 4 - Loopback for further explanation.

Section 2 can be executed with the Test Hood which is used by the selftest to also check the Z-80 Serial Communications Controller circuits, line drivers, and line receivers. The Test Hood is part of the CIO Service Kit available to HP Customer Engineers. Refer to Section 5 - Selftest for further information regarding the Test Hood and the selftest itself. If selftest fails, the following message is output:

```
*** SELFTEST OF MUX CARD FAILED. (MUXERR 5004)
```

If the `MUXDIAG` selftest fails, replace the MUX card. For removal and replacement instructions, refer to the the hardware reference manual corresponding to your MUX. Remember to remove the EPROM from the bad card to place it on the replacement board.

When running under HP-UX, if `MUXDIAG` can not find certain system administration files, the download fails with one of the following messages output (also refer to "Error Messages" at the end of this section):

```
*** DOWNLOAD TO MUX RAM FAILED
    COULD NOT OPEN /etc/file
    AS HP-UX STDIN (DSSERR 2400)
```

where "file" is a download file in the /etc directory.

```
*** DOWNLOAD TO MUX RAM FAILED
    COULD NOT DUP /dev/diag/mux
    AS HP-UX STDOUT, FILE DESCRIPTOR WAS
    fd (DSSERR 2401)
```

where `/dev/diag/mux*` is a MUX card device file name in the `/dev/diag` directory, and "fd" is the file descriptor associated with `/dev/diag/mux*`.

```
*** DOWNLOAD TO MUX RAM FAILED
    HP-UX COULD NOT EXEC /etc/download (DSSERR 2402)
```

The `/etc/download` file could not be executed.

Refer to the *HP-UX System Administrator Manual* for file information. Some reasons for not finding or not executing an HP-UX file are:

File does not exist Either the file exists in the wrong directory or it has been renamed accidentally. Contact your System Administrator to install the missing file in the correct directory.

Incorrect permissions Use the HP-UX `ls` or `ll` commands to check the file's permissions. If read/write/execute permissions have been disabled, contact your System Administrator. The System Administrator can change the file permissions.

If the system is having problems, either troubleshoot the system or reboot. Refer to the *HP-UX System Administrator's Manual* and to the Troubleshooting Chapter of the computer's *Hardware Support Manual*.

An internal `MUXDIAG` error can cause one of the following messages:

```
*** DOWNLOAD TO MUX RAM FAILED
    HP-UX PROCESS EXIT STATUS =
    nnnn (DSSERR 2403)
```

The value "nnnn" is the process exit status returned by the program `/etc/download`.

```
*** DOWNLOAD TO MUX RAM FAILED
    HP-UX ABNORMAL PROCESS EXIT STATUS =
    nnnn (DSSERR 2404)
```

The value "nnnn" is the abnormal process exit status returned by the program `/etc/download`.

Try running `MUXDIAG` again before calling the HP Sales and Service Office.

Section 3 - IDENTIFY

This section displays identification information about the MUX card. Section 3 is also one of the default sections that is executed if no parameters are specified in `RUN MUXDIAG`.

SAMPLE OUTPUT MESSAGE

```
Beginning Section 3
   HP27140A/HP98196A 6 port mux - Card ID = 7
   Firmware ID = 3
   Firmware Revision = 2939
   Hardware Revision = 0
   Active Ports = 6
   Multiplexing Support Code = 1 - Logchannel Only
   Parity Support = 0 - Not Available
   Data Transfer Mode = 2 - Word and Byte
End of Section 3
```

where the values or messages are defined as follows:

a,b,ccc,d	These numeric values identify the MUX card and are helpful in finding out the "age" of the card.
`Active Ports`	If 6, 8, or 16, then the MUX card is downloaded. (HP-UX Only) If 1, then the MUX card is not downloaded. Run Section 2 - CLEAR test to download the RAM code to the MUX card. Under MPE/iX, this number may vary and has no significance to the RAM code downloading process.
`Multiplexing Support Code`	The channel architecture supports two protocols with which I/O cards can communicate with the channel. The two protocols are subchannel and logchannel. However, the MUX card uses its own unique protocol. The current release of the downloaded firmware incorrectly returns: `Multiplexing Support Code 1 - Logchannel Only`.
`Parity Support`	Parity data checking between the backplane and the channel is NOT supported. However, parity data checking is supported at the frontplane which is what the HP-UX MUX driver uses to check parity.
`Data Transfer Mode`	Data may be transferred either in one or two byte quantities, or both.

Section 4 - LOOPBACK

This section tests data path between the channel and the device through the card. The test is performed by writing and reading a fixed data pattern. Either one or both of the following two steps can be executed. Step 40 is one of the default steps that is executed if no parameters are specifies in `RUN MUXDIAG`.

SAMPLE OUTPUT MESSAGE

```
Beginning Section 4

WARNING: If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 40
   BACKPLANE loopback test completed successfully.
End of Step 40
Beginning Step 41
   FRONTPLANE loopback test completed successfully.
End of Step 41
Beginning Step 42
   This function is not available on this card.
End of Step 42
Beginning Step 43
   This function is not available on this card.
End of Step 43
End of Section 4
```

Step 40 **Backplane Loopback:** tests the back interface circuitry, which handles all communication to the CIO backplane. This test is non-destructive (in normal mode) and can be run while data is being sent to and from the MUX card.

Step 41 **Frontplane Loopback:** tests the front plane interface circuitry, which includes the RS-232 receivers and line drivers. This test could destroy user data being sent from the MUX driver and is therefore a destructive test. Users must complete their terminal I/O before running this test.

Step 42 **Cable Loopback:** tests the cable from the card to the panel (ADP). This test could destroy user data being sent from the MUX driver and is, therefore, a destructive test. Users must complete their terminal I/O before running Step 42.

Step 43 **Panel Loopback:** tests both panels' (ADP) circuitry. This test could destroy user data being sent from the MUX driver and is, therefore, a destructive test. Users must complete their terminal I/O before running Step 43.

The MUX card must be downloaded before running the frontplane loopback test. If the RAM code is not downloaded in the MUX card, the following messages occurs:

```
***WARNING - FRONTPLANE LOOPBACK TEST WAS SKIPPED BECAUSE THE CARD
FIRMWARE WAS NOT LOADED (DSSERR 2405)
```

To assure that the MUX card is downloaded, run the CLEAR section first, especially when running these test sections in a loop (for example, **RUN MUXDIAG** pdev=4.3 loopcount=2 sec=1,2,4 step=40,41).

Note Some MUX cards may not have cable and/or panel loopback.

If a loopback test fails, the following error messages are displayed depending on the test executed:

```
*** ERROR IN section 4
*** ERROR IN STEP 40
*** BACKPLANE LOOPBACK TEST FAILED.  (MUXERR 5003)
    FAILURE CODE =
    DATA =
    PORT =
*** ERROR IN section 4
*** ERROR IN STEP 41
*** FRONTPLANE LOOPBACK TEST FAILED.  (MUXERR 5003)
    FAILURE CODE =
    DATA =
    PORT =
```

If the FAILURE CODE is 1, the MUX card is bad. Refer to the hardware reference manual corresponding to your MUX for instructions on removing and replacing the MUX card. Remember to remove the EPROM from the bad card and place it on the replacement board.

The following message is also displayed if the FAILURE CODE is 2 for either loopback test:

```
*** I/O FAILURE OCCURRED TRYING TO DO LOOPBACK (MUXERR 5002)
```

The MUX card may be bad. Run the channel diagnostics. Check the device access permissions by running Sections 2, 5, and 6. If these sections execute without internal errors and if the loopback FAILURE CODE continues to be 2, the MUX card is bad. Replace the card.

Section 5 - SELFTEST

This section executes the selftest program stored in the EPROM chip on the MUX card. The test operates on a GO/NO-GO basis and determines if the card will reliably pass data in both directions through the circuitry.

The selftest can be executed with or without the Test Hood. The Test Hood is part of the CIO Service Kit available to HP Customer Engineers.

Without the Test Hood, the selftest verifies the following:

- On-board circuitry of the EPROM
- RAM
- Operation of the Z-80 microprocessor
- Resident DMA operation
- Timer chip performance

With the Test Hood, the following items are tested in addition to the above items:

- Z-80 Serial Communications Controller (SCC) circuits
- Line drivers
- Line receivers

Selftest (with or without the Test Hood) does not validate the backplane circuitry. The backplane circuitry can be tested in Section 4, Loopback, described above.

If no errors occur, `MUXDIAG` outputs the following messages:

```
Beginning Section 5

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

   Selftest of MUX Card completed successfully.
End of Section 5
```

If selftest fails, the following message is output:

```
*** SELFTEST OF MUX CARD FAILED. (MUXERR 5004)
```

If the I/O channel fails and `MUXDIAG` cannot function, a LED mounted on the board may be used to interpret the test results. Without the hood, the LED lights up for approximately 2 seconds if the test was executed due to a RESET condition. The DEVICE CLEAR/DEVICE ENABLE sequence turns the LED on for 13 to 15 seconds. If the LED fails to go off, the selftest has failed.

With the optional Test Hood, a second LED mounted on the hood also shows whether the tests were successful. Using the Test Hood does not change the length of time that the selftest takes to run. If the LED fails to go off, the selftest is a NO-GO and has failed.

If the `MUXDIAG` selftest fails, replace the MUX card.

Refer to the hardware reference manual corresponding to your MUX for instructions on removing and replacing the MUX card. Remember to remove the EPROM from the bad card and place it on the replacement board.

Section 6 - WRITE/READ

This section writes data to the MUX RAM, reads it back, and then compares the data.

SAMPLE OUTPUT MESSAGE

```
Beginning Section 6

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 60
   Write/Read test of MUX card completed successfully.
End of Step 60
Beginning Step 61
   Read test of MUX card completed successfully.
End of Step 61
Beginning Step 62
   Write/Read compare test completed successfully.
End Step 62
End of Section 6
```

The following steps are available with this test section:

Step 60	**RAM write test**
Step 61	**RAM read test**
Step 62	**Compare RAM write and read test**

If this test fails, one of the following error messages may occur. Also, replace the MUX card.

Refer to the hardware reference manual corresponding to your MUX for instructions on removing and replacing the MUX card. Remember to remove the EPROM from the bad card and place it on the replacement board.

```
*** WRITE/READ TEST FAILED TO WRITE ENOUGH DATA.  (MUXERR 5005)
WROTE xxx BYTES, BUT SHOULD HAVE WRITTEN yyy BYTES.

*** WRITE/READ TEST FAILED TO READ ENOUGH DATA.  (MUXERR 5006)
READ xxx BYTES, BUT SHOULD HAVE READ yyy BYTES.

*** WRITE/READ TEST OF MUX CARD FAILED.  (MUXERR 5007)
BYTE AT BUFFER x WAS y BUT SHOULD HAVE BEEN z.
```

Section 9 - DUMP

This section dumpts the MUX card RAM to a specified file. Section 9 is not a default section, and there are no steps.

SAMPLE OUTPUT MESSAGE:

```
Beginning Section 9
 DUMP completed successfully.  The file DUMP9008310732 contains the binary listing.
End of Section 9
```

Examples

Example 1: Running the Default Sections

The following example runs the default sections 1, 3, and 4 (Step 40).

```
DUI > mode sum
Single User Mode (SUM).
DUI > run muxdiag pdev=4.3

     **********************************************************
     *****      HP CIO Asynchronous Port Multiplexer       ***
     *****                                                 ***
     *****                 (MUX) Diagnostic                ***
     *****                                                 ***
     *****     (c) Copyright Hewlett-Packard Company 1987  ***
     *****              All Rights Reserved.               ***
     *****                                                 ***
     *****                 Version A.03.00                 ***
     *****                                                 ***
     **********************************************************

     Welcome, Today is Thursday, January 4, 1990 at 10:50:30.

   Beginning Section 1
   Beginning Step 10
      The MUX card RAM code is LOADED.
   End of Step 10
   Beginning Step 11
      The MUX card is marked as NOT BROKEN.
   End of Step 11
   End of Section 1
   Beginning Section 2

   WARNING:  If the diagnostic is aborted during this section, the card
   may be left in a corrupted state.  If this is the case, run section 2 to
   reset the card.

      The MUX card CLEARED successfully.
   End of Section 2
   Beginning Section 3
      HP27140A/HP98196A 6 port mux - Card ID = 7
      Firmware ID = 3
      Firmware Revision = 2939
      Hardware Revision = 0
      Active Ports = 6
      Multiplexing Support Code = 1 - Logchannel Only
      Parity Support = 0 - Not Available
      Data Transfer Mode = 2 - Word and Byte
   End of Section 3
```

```
Beginning Section 4

WARNING: If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 40
   BACKPLANE loopback test completed successfully.
End of Step 40
End of Section 4
Beginning Section 6

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 61
   Read test of MUX card completed successfully.
End of Step 61
MUXDIAG Exiting...
MUXDIAG has terminated
```

Example 2: Running MUXDIAG Twice

The following example runs MUXDIAG twice with loop=2.

```
DUI > mode sum
Single User Mode (SUM).
DUI >run muxdiag pdev=4.3 loop=2

     ***********************************************************
     *****      HP CIO Asynchronous Port Multiplexer        ***
     *****                                                  ***
     *****                  (MUX) Diagnostic                ***
     *****                                                  ***
     *****     (c) Copyright Hewlett-Packard Company 1987   ***
     *****                All Rights Reserved.              ***
     *****                                                  ***
     *****                  Version A.03.00                 ***
     *****                                                  ***
     ***********************************************************

     Welcome, Today is Thursday, January 4, 1990 at 10:50:30.

   Loop count = 1
   Beginning Section 1
   Beginning Step 10
      The MUX card RAM code is LOADED.
   End of Step 10
   Beginning Step 11
      The MUX card is marked as NOT BROKEN.
   End of Step 11
   End of Section 1
   Beginning Section 2

   WARNING:  If the diagnostic is aborted during this section, the card
   may be left in a corrupted state.  If this is the case, run section 2 to
   reset the card.

      The MUX card CLEARED successfully.
   End of Section 2
   Beginning Section 3
      HP27140A/HP98196A 6 port mux - Card ID = 7
      Firmware ID = 3
      Firmware Revision = 2939
      Hardware Revision = 0
      Active Ports = 6
      Multiplexing Support Code = 1 - Logchannel Only
      Parity Support = 0 - Not Available
      Data Transfer Mode = 2 - Word and Byte
   End of Section 3
```

```
Beginning Section 4

WARNING: If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 40
      BACKPLANE loopback test completed successfully.
   End of Step 40
End of Section 4
Beginning Section 6

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 61
   Read test of MUX card completed successfully.
End of Step 61
Loop count = 2    <-----------------------start of second loop
Beginning Section 1
Beginning Step 10
   The MUX card RAM code is LOADED.
End of Step 10
Beginning Step 11
   The MUX card is marked as NOT BROKEN.
End of Step 11
End of Section 1
Beginning Section 2

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

   The MUX card CLEARED successfully.
End of Section 2
Beginning Section 3
   HP27140A/HP98196A 6 port mux - Card ID = 7
   Firmware ID = 3
   Firmware Revision = 2939
   Hardware Revision = 0
   Active Ports = 6
   Multiplexing Support Code = 1 - Logchannel Only
   Parity Support = 0 - Not Available
   Data Transfer Mode = 2 - Word and Byte
End of Section 3
```

```
Beginning Section 4

WARNING: If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 40
   BACKPLANE loopback test completed successfully.
End of Step 40
End of Section 4
Beginning Section 6

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 61
   Read test of MUX card completed successfully.
End of Step 61
MUXDIAG Exiting...
MUXDIAG has terminated
```

Example 3: MUXDIAG Sections 1, 2, 3, 5, and 6

The following example runs MUXDIAG sections 1, 2, 3, 5, and 6.

```
DUI > mode sum
Single User Mode (SUM).
DUI > run muxdiag pdev=4.3 sec=1,2,3,5,6

     **********************************************************
     *****       HP CIO Asynchronous Port Multiplexer      ***
     *****                                                 ***
     *****                (MUX) Diagnostic                 ***
     *****                                                 ***
     *****   (c) Copyright Hewlett-Packard Company 1987    ***
     *****              All Rights Reserved.               ***
     *****                                                 ***
     *****                 Version A.03.00                 ***
     *****                                                 ***
     **********************************************************

     Welcome, Today is Thursday, January 4, 1990 at 10:50:30.

   Beginning Section 1
   Beginning Step 10
      The MUX card RAM code is LOADED.
   End of Step 10
   Beginning Step 11
      The MUX card is marked as NOT BROKEN.
   End of Step 11
   End of Section 1
   Beginning Section 2

   WARNING:  If the diagnostic is aborted during this section, the card
   may be left in a corrupted state.  If this is the case, run section 2 to
   reset the card.

      The MUX card CLEARED successfully.
   End of Section 2
   Beginning Section 3
      HP27140A/HP98196A 6 port mux - Card ID = 7
      Firmware ID = 3
      Firmware Revision = 2939
      Hardware Revision = 0
      Active Ports = 6
      Multiplexing Support Code = 1 - Logchannel Only
      Parity Support = 0 - Not Available
      Data Transfer Mode = 2 - Word and Byte
   End of Section 3
```

```
Beginning Section 5

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

   Selftest of MUX Card completed successfully.
End of Section 5
Beginning Section 6

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 61
   Read test of MUX card completed successfully.
End of Step 61
MUXDIAG Exiting...
MUXDIAG has terminated
```

Example 4: MUXDIAG Sections 1, 3, 4 (steps 41, 42, 43), 5, and 6

The following example runs MUXDIAG sections 1, 3, 4 (steps 41, 42, 43), 5, and 6.

```
DUI > mode sum
Single User Mode (SUM).
DUI > run muxdiag pdev=4.3 sec=1,3,4,5,6 step=41,42,43

    ***********************************************************
    *****     HP CIO Asynchronous Port Multiplexer        ***
    *****                                                 ***
    *****                (MUX) Diagnostic                 ***
    *****                                                 ***
    *****    (c) Copyright Hewlett-Packard Company 1987   ***
    *****               All Rights Reserved.              ***
    *****                                                 ***
    *****                Version A.03.00                  ***
    *****                                                 ***
    ***********************************************************

    Welcome, Today is Thursday, January 4, 1990 at 10:50:30.

   Beginning Section 1
   Beginning Step 10
      The MUX card RAM code is LOADED.
   End of Step 10
   Beginning Step 11
      The MUX card is marked as NOT BROKEN.
   End of Step 11
   End of Section 1
   Beginning Section 2

   WARNING:  If the diagnostic is aborted during this section, the card
   may be left in a corrupted state.  If this is the case, run section 2 to
   reset the card.

      The MUX card CLEARED successfully.
   End of Section 2
   Beginning Section 3
      HP27140A/HP98196A 6 port mux - Card ID = 7
      Firmware ID = 3
      Firmware Revision = 2939
      Hardware Revision = 0
      Active Ports = 6
      Multiplexing Support Code = 1 - Logchannel Only
      Parity Support = 0 - Not Available
      Data Transfer Mode = 2 - Word and Byte
   End of Section 3
```

```
Beginning Section 4

WARNING: If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 41
   FRONTPLANE loopback test completed successfully.
End of Step 41
Beginning Step 42
   This function is not available on this card.
End of Step 42
Beginning Step 43
   This function is not available on this card.
End of Step 43
End of Section 4
Beginning Section 6

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 61
   Read test of MUX card completed successfully.
End of Step 61
MUXDIAG Exiting...
MUXDIAG has terminated
```

Example 5: MUXDIAG Input and Output Redirected

The following example runs **MUXDIAG** with its input and output redirected. The output is redirected to a file called **muxdiag.out**. This example shows how to run **MUXDIAG** with only one MUX card. As superuser and in the C shell (**csh**), type the following commands. The **#** prompt is the shell prompt.

```
# csh                                    <---- go into C shell
echo" mode sum \
run muxdiag pdev=4.3" | sysdiag >& /tmp/muxdiag.out  <---- no # appears
# cat /tmp/muxdiag.out                       <---- displays the following

     **********************************************************
     *****      HP CIO Asynchronous Port Multiplexer       ***
     *****                                                 ***
     *****                 (MUX) Diagnostic                ***
     *****                                                 ***
     *****     (c) Copyright Hewlett-Packard Company 1987  ***
     *****                All Rights Reserved.             ***
     *****                                                 ***
     *****                 Version A.03.00                 ***
     *****                                                 ***
     **********************************************************
     Welcome, Today is Thursday, January 4, 1990 at 10:50:30.

DUI > mode sum
Single User Mode (SUM).
DUI > run muxdiag pdev=4.3

     **********************************************************
     *****      HP CIO Asynchronous Port Multiplexer       ***
     *****                                                 ***
     *****                 (MUX) Diagnostic                ***
     *****                                                 ***
     *****     (c) Copyright Hewlett-Packard Company 1987  ***
     *****                All Rights Reserved.             ***
     *****                                                 ***
     *****                 Version A.03.00                 ***
     *****                                                 ***
     **********************************************************

     Welcome, Today is Thursday, January 4, 1990 at 10:50:30.
```

```
Beginning Section 1
Beginning Step 10
   The MUX card RAM code is LOADED.
End of Step 10
Beginning Step 11
   The MUX card is marked as NOT BROKEN.
End of Step 11
End of Section 1
Beginning Section 2

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

   The MUX card CLEARED successfully.
End of Section 2
Beginning Section 3
   HP27140A/HP98196A 6 port mux - Card ID = 7
   Firmware ID = 3
   Firmware Revision = 2939
   Hardware Revision = 0
   Active Ports = 6
   Multiplexing Support Code = 1 - Logchannel Only
   Parity Support = 0 - Not Available
   Data Transfer Mode = 2 - Word and Byte
End of Section 3
Beginning Section 4

WARNING: If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 40
   BACKPLANE loopback test completed successfully.
End of Step 40
End of Section 4
Beginning Section 6

WARNING:  If the diagnostic is aborted during this section, the card
may be left in a corrupted state.  If this is the case, run section 2 to
reset the card.

Beginning Step 61
   Read test of MUX card completed successfully.
End of Step 61
MUXDIAG Exiting...
MUXDIAG has terminated
```

Error Messages

This section lists all the error messages that may be generated by **MUXDIAG** along with brief explanations. The messages are listed in numerical order. Listed below each error message are probable cause and recommended action statement. The actual cause and action may differ from this list depending upon the particular circumstances of a given situation.

Error messages without the **MUXERR #** or **DSSERR #** trailer are generated by the Online Diagnostic Subsystem or the operating system. For errors outside of **MUXDIAG**, consult the DUI section of this manual and the operating system manuals.

The following error messages with the **DSSERR #** trailer are displayed by **MUXDIAG**.

5001 **INVALID MUX ID !. (MUXERR 5001)**
CAUSE When an ID request to the card was done it did not match a known card ID.
ACTION The MUX card may be bad. Try running the channel diagnostics. Check the device permissions by running sections 2, 5, and 6. If these tests pass but this error continues, the card is bad. Refer to the hardware reference manual for your MUX for instructions on removing and replacing the MUX card.

5002 ***** I/O FAILURE OCCURRED TRYING TO DO LOOPBACK (MUXERR 5002)**
CAUSE This message occurs when the FAILURE CODE is 2 on any loopback test from Section 4.
ACTION The MUX card may be bad. Try running the channel diagnostics. Check the device permissions by running sections 2, 5, and 6. If these tests pass but this error continues, the card is bad. Refer to the hardware reference manual for your MUX card for instructions on removing and replacing the MUX card.

5003
```
***  ! LOOPBACK TEST FAILED (MUXERR 5003)
         FAILURE CODE = !
          DATA = !
           PORT = !
```
CAUSE The specified loopback test failed.
ACTION The MUX card may be bad. Try running the channel diagnostics. Check the device permissions by running sections 2,5, and 6. If these tests pass, but this error continues, the card is bad. Refer to the hardware reference manual for your MUX for instructions onr removing and replacing the MUX card.

5004 ***** SELFTEST OF MUX CARD FAILED (MUXERR 5004)**
CAUSE The section 5 selftest failed.
ACTION The MUX card is bad and needs to be replaced. Refer to the hardware reference manual for your MUX for instructions on removing and replacing the MUX card.

5005 `*** WRITE/READ TEST FAILED TO WRITE ENOUGH DATA (MUXERR 5005)`
`WROTE ! BYTES, BUT SHOULD HAVE WRITTEN ! BYTES.`

CAUSE The test failed to write the correct number of bytes to the card.

ACTION The MUX card is bad and needs to be replaced. Refer to the hardware reference manual for your MUX for instructions on removing and replacing the MUX card.

5006 `*** WRITE/READ TEST FAILED TO READ ENOUGH DATA (MUXERR 5006)`
`READ ! BYTES, BUT SHOULD HAVE READ ! BYTES.`

CAUSE The test failed to read the correct number of bytes from the card.

ACTION The MUX card is bad and needs to be replaced. Refer to the hardware reference manual for your MUX for instructions on removing and replacing the MUX card.

5007 `*** WRITE/READ TEST OF MUX CARD FAILED (MUXERR 5007)`
`BYTE AT BUFFER [!] WAS !, BUT SHOULD HAVE BEEN !.`

CAUSE The byte written to the card did not match the bytes that were read from the card.

ACTION The MUX card is bad and needs to be replaced. Refer to the hardware reference manual for your MUX for instructions on removing and replacing the MUX card.

5020 `*** THE RAM DUMP FAILED. (MUXERR 5020)`

CAUSE The MUX RAM DUMP test failed.

ACTION Check to make sure there is enough disk space to write the file. Also check for write access to the default directory. If the problem continues, the MUX card is bad and needs to be replaced. Refer to the hardware reference manual for your MUX for instructions on removing and replacing the MUX card.

8

Asynchronous Muliplexer Diagnostic

Introduction

The Asynchronous Multiplexer Diagnostic (PMUXDIAG) provides the ability to detect and isolate hardware failures in the following HP-PB Multiplexer products:

- HP 40299A/B, HP J2092A, HP J2093A and HP J2094A Asynchronous Multiplexer cards
- The Multiplexer portion of the multifunction I/O cards for the HP 9000 Series 8x7 (A1751-60001, A1703-60004, A1703-60022).

Failure isolation is to one of the products field replaceable units (FRUs):

HP 40299A/B :

- 8 channel Mux card
- Mux card to Active Distribution Panel (ADP) Cable
- Active Distribution Panel (ADP)

Note

For the 40299A with RS-232 ADP (part no. 40299-60002) or 40299B with RS-422 ADP (part no. 5062-3085), the panel loopback requires a loopback hood (part no. 5181-2030) for each port under test.

HP J2092A :

- 16 channel direct connect mux card

HP J2093A :

- 16 channel direct connect mux card

Note

You will need a 78 pin loopback hood (part no. J2093-60002) to be able to fully test the Mux card alone using the panel loopback. If you wish to test the cables and Direct Connection Panels (DDPs), run the panel loopback test on the port to be tested and ensure a 25 pin loopback hood (part no. 5181-2030) is installed on that port of the DDP.

HP J2094A :

- 16 channel modem connect mux card
- Mux card to Active Distribution Panels (ADPs) cable
- Active Distribution Panels (ADPs)

A1751-60001 :

- 8 channel Mux portion of the Multifunction I/O card

A1703-60004 :

- 8 channel Mux portion of the Multifunction I/O card
- Mux card to Active Distribution Panel (ADP) Cable
- Active Distribution Panel (ADP)

A1703-60022 :

- 16 channel Mux portion of the Multifunction I/O card
- Mux card to Active Distribution Panel (ADP)/Direct Distribution Panel (DDP) Cable
- Active Distribution Panel (ADP)

Note

The testing of ports 1 to 7 (on the ADP) is as for the AP-Mux Card (40299B) - backplane, frontplane, and panel loopbacks. When testing ports 8 to 15, which are on the Direct Distribution Panel (DDP), backplane and frontplane have the same function, and a panel loopback loops back just before the 78 pin connector on the card.

Although this diagnostic can better test the above mentioned FRUs if HP terminals are connected to the HP-PB MUX card's ports (providing a final "loopback" path to/from the distribution panel), this diagnostic will not test devices connected to the card.

Several of the tests (defined later in the Section Functional Descriptions) are capable of destroying user data associated with the HP-PB MUX card under test, or the state and memory of the HP-PB MUX card itself. Therefore care must be used when executing this diagnostic on-line in a multiuser environment, limiting testing to unused HP-PB MUX cards or unused ports associated with the HP-PB MUX card under test.

Defects and Enhancements

Submit defect reports and enhancement requests concerning this diagnostic through the STARS database referencing product number 30600-10030.

Minimum Configuration

- A PA-RISC computer system with enough basic hardware to boot and execute the HP-UX operating system
- A terminal connected to the system console port (or another HP-PB MUX card in the system)
- An HP-PB Multiplexer Card

The components above constitute the minimum configuration required to run **PMUXDIAG**. For further testing, to take advantage of the loopback capabilities of **PMUXDIAG**, you will also need one of the following (depending upon your configuration):

- **HP J2092A**: Mux card only.
- **HP 40299A/B** and **A1703-60004**:
  - Mux card (or Mux portion of Multifunction I/O card) to Active Distribution Panel (ADP) Cable
  - Active Distribution Panel (ADP)
- **HP J2093A:**
  - 78 pin loopback hood (part no. J2093-60002)

Note

You will need this 78 pin loopback hood to be able to fully test the Mux card, using the panel loopback. If you wish to test the cables and Direct Connection Panels (DDPs), run the panel loopback test on the port to be tested and ensure a 25 pin loopback hood (part no. 5181-2030) is installed on the corresponding port of the DDP.

- **HP J2094A**:
  - Mux card to Active Distribution Panels (ADPs) cable
  - 2 Active Distribution Panels (ADPs)
- **A1751-60001**: Mux portion of the Multifunction I/O card only.
- **A1703-60022**:
  - Mux card to Active Distribution Panel (ADP)/Direct Distribution Panel (DDP) Cable
  - 1 Active Distribution Panel (ADP)

Any port connected to the system, or virtual terminal connection over a LAN, may be used to run this diagnostic. From port X on MUX Y it is possible to test all HP-PB MUX cards in the system (excluding MUX Y).

In order to test the HP-PB MUX card up to the junction panel connectors, the diagnostic user may also be prompted to attach a loopback hood (part no. 5181-2030) to the junction panel, if the active distribution panel is a version without electronic loopback.

In order to execute this diagnostic the following software must be available:

- HP-UX operating system (Version 8.02 or later)
- `PMUXDIAG` (Version A.03.00 or later), and its associated Diagnostic Access Routine (DAR), executing under the Online Diagnostics Subsystem.
- The operating system driver for the HP-PB Multiplexer Cards
- The downloadable microcode for the HP-PB Multiplexer Cards

Operating Instructions

PMUXDIAG is accessed by the user via the Diagnostic User Interface (DUI).

Default Tests

The default sections and steps for this diagnostic are:

```
Section 1    State

Section 3    Identify

Section 4    Loopback

  Step 420-435       Frontplane Loopback
```

Note Note that the command below will run the default sections, listed above.

```
DUI>pmuxdiag ldev=<logical MUX specifier>
```

RUN Command

To bring up the Online Diagnostic subsystem, enter the following command at the system prompt:

```
sysdiag
```

The diagnostic subsystem responds with the following prompt indicating that access has been granted to the user:

```
DUI >
```

Typing help causes a summary of the DUI function and its commands to appear on the screen.

After the Online Diagnostics Subsystem has been started, this diagnostic can be executed using the command:

```
DUI>pmuxdiag ldev=<logical MUX specifier>
```

All Online Diagnostic Subsytem run string parameters are acceptable when executing this diagnostic. For example,

```
DUI>pmuxdiag ldev=<logical MUX specifier> sc 4 (420/423)
```

will run loopback (Section 4), Steps 420 to 423, corresponding to Frontplane loopback for the circuitry associated with ports 0 to 3.

Test Execution

When the diagnostic is first started, a header and welcome message is displayed.

Detailed Test Descriptions

The following discussion describes the testing and actions performed by each of the diagnostic's sections and steps, the optimum order in which they should be executed, and any message output that has special significance. The following test sections and steps are available with this diagnostic:

```
Online Diagnostic
Function                            Section         Steps

STATE                                  1

CLEAR                                  2
    Common MUX Logic                                  199
    Port specific logic                               200...215

IDENTIFY                               3

LOOPBACK                               4
  Backplane
    Common MUX logic                                  399
    Port specific logic                               400...415
  Frontplane
    Common MUX logic                                  419
    Port specific logic                               420...435
  Panel
    Port specific logic                               460...475
  Terminal
    Port specific logic                               480...495

SELFTEST                               5
    Tests the whole MUX

RESET                                  7
    Port specific logic                               700...715

HALT                                   8

DUMP                                   9
    Global dump                                       900
    Partial dump                                      901
```

Section 1 - STATE

This section permits the user to display the current state of the HP-PB MUX card or a selected port. Results provided are:

- Broken specifies that the driver sees the HP-PB MUX as defective, for instance, when there is a hardware failure.
- RAM is loaded indicates that the HP-PB MUX has the firmware running on it.

WARNINGS and CAUTIONS:

None - there are no destructive effects to users connected to the HP-PB MUX when this section is executed.

Section 2 - CLEAR

This section permits the user to completely initialize the entire HP-PB MUX card (including a download of microcode), or completely initialize a selected HP-PB MUX port.

Step 199 This step resets and initializes the entire HP-PB MUX card. First the hardware is reset and the board is tested; next, the microcode is downloaded and started. After executing this step the card will be set to its initial state.

Steps 200-215 These steps perform a complete initialization on the selected HP-PB MUX port (steps 200-215 correspond to ports 0-15) without disturbing the other HP-PB MUX ports. A complete initialization is defined to be:

a. Reset the selected port's hardware

b. Reset the selected port's associated microcode

c. Reset the port's associated driver software

These steps require that the microcode be loaded in order to function correctly.

WARNINGS and CAUTIONS:

When step 199 is executed, any user connected to any port on the HP-PB MUX may lose data, be locked up, or may be disconnected. Any user connected to the selected port on the HP-PB MUX when steps 200-215 are executed may lose data, be locked up, or may be disconnected.

Section 3 - IDENTIFY

This section displays current information about the HP-PB MUX card and ports under test. It is not necessary that microcode be loaded to use this section. There are no destructive effects to users connected to the HP-PB MUX when this section is executed.

- The firmware/ROM field shows whether the firmware or ROM interpreter is active.
- Next, The revision of the ROM or firmware is displayed, depending on which one is active.
- To check the validity of returned information a checksum is performed with the pertinent IODC information. If the checksum fails, an error message is displayed.
- The type of module (IODC information) is checked. If it is not the one expected, an error message is displayed.

The processor to HP-PB MUX path, as well as some basic HP-PB MUX hardware, is known to be partially functional if this step completes successfully .

WARNINGS and CAUTIONS:

None - there are no destructive effects to users connected to the HP-PB MUX when this section is executed.

Section 4 - LOOPBACK

This section provides several associated loopback steps which test various data paths from the system processor up to the MUX ports or, if an HP terminal is connected, the attached terminal.

The purpose of this section is to help detect and isolate hardware failures to the faulty FRU: the MUX card (or MUX part of the multifunction I/O card) and, depending on the product, the connection cable, the MUX Card to ADP or DDP cable, the ADP (or ADPs) and the DDPs.

Therefore testing is done by increasing the hardware covered as the step numbers increase (i.e., "onion peel" testing):

Backplane Loopback	Steps 399-415 test a small amount of the of the MUX's HP-PB interface logic.
Frontplane Loopback	Steps 419-435 test a fairly large amount of MUX logic
Panel Loopback	Steps 460-475 test the data path up to the terminal connector (25 pin).
Terminal Loopback	Using the entire Mux card and ADP Panel(s) if they exist, Steps 480-495 test all composite hardware under system "usage conditions", by exploiting the loopback feature in HP terminals.

Note Terminal loopback tests will only execute properly when the port(s) under test are connected to HP terminals.

Additional information about the hardware condition of the HP-PB MUX card itself can be obtained by executing the SELFTEST Section.

WARNINGS and CAUTIONS:

When any step in Section 4 is executed, devices connected to any port on the HP-PB Mux under test may loose data, be locked up or disconnected.

Backplane Loopback

This set of Section 4 steps tests a data path from the main processor to the HP-PB backplane interface logic associated with the entire MUX card, or with a selected port. Note that MUX microcode must be loaded for this section's steps to execute properly.

Step 399 This step tests backplane MUX logic common to all ports: the Supervisor Element registers, the I/O Element registers, and the interface chip FIFOs which handle communications between the HP-PB backplane and the main processor.

Steps 400-415 These steps test main processor to HP-PB backplane interface logic and FIFOs unique to a specific port (steps 400-415 correspond to ports 0-15).

Frontplane Loopback

This set of Section 4 steps tests a data path from the main processor to the HP-PB MUX frontplane interface logic (i.e., the interface to the distribution panel) associated with the entire MUX card, or with a selected port. Note that MUX microcode must be loaded for this section's steps to execute properly.

Step 419 This step tests main processor through MUX frontplane communication logic, common to all ports.

Steps 420-435 These steps test main processor through MUX frontplane communication logic, specific to a selected port (steps 420-435 correspond to ports 0-15).

Panel Loopback

These Section 4 steps test a data path for a selected port from the HP-PB MUX card up to the terminal connector. Note that MUX microcode must be loaded for this section's steps to execute properly.

Steps 460-475 These steps test a data path consisting of:

- the main processor
- the HP-PB bus
- MUX card logic common to all ports
- the MUX card to panel connection cable (if applicable)
- distribution panel logic common to all ports (if applicable)
- panel logic specific to the selected port (steps 460-475 correspond to ports 0-15).

Note that some versions of the distribution panel require that the user install a panel loopback hood. When this is the case, which the diagnostic will detect, the user will be asked to install this hood via a special message.

Note

With the J2092A, J2093A and ports 8-15 of the A1703-60022, the panel loopback tests up to the 78 pin Mux connector only. In the case of ports 2-7 of the A1751-60001, the panel loopback tests up to the hexapus cable connector. Additionally, the J2093A requires a 78 pin loopback hood (part no. J2093-60002) to execute this loopback.

Terminal Loopback

These Section 4 steps test a data path for a selected port from the HP-PB MUX card up through a terminal, as long as that terminal is an HP terminal. Note that this part of the diagnostic is not intended to test the terminal, or the terminals software driver. Rather it is intended to exploit loopback capabilities in the terminal to more fully diagnose the FRUs under test. Note that MUX microcode must be loaded for this section's steps to execute properly.

Steps 480-495 These steps test a data path consisting of:

- the main processor
- the HP-PB bus
- MUX card logic common to all ports
- the HP-PB MUX card to panel connection cable (if applicable)
- distribution panel logic common to all ports (if applicable)
- panel logic specific to the selected port (steps 480-495 correspond to ports 0-15), only when an HP terminal is connected to the selected port.

Section 5 - SELFTEST

This section permits the user to execute a selftest of the HP-PB MUX Card. The selftest is embedded in the ROM resident on the HP-PB MUX card under test.

All error messages specific to this section indicate that the card itself is bad.

WARNINGS and CAUTIONS:

When any of this section is executed, any user connected to any port on the HP-PB MUX, may lose data, be locked up, or may be disconnected.

Section 7 - PORT RESET

This section permits the user to perform a partial initialization on a selected HP-PB MUX port.

Steps 700-715 These steps perform a partial initialization on the selected HP-PB MUX port (steps 700-715 correspond to ports 0-15), without disturbing the other HP-PB MUX ports. A partial initialization will do the following:

a. Reset the selected port's hardware

b. Reset the selected port's associated microcode

This step will not work when HP-PB MUX microcode is not loaded. This step will fail if executed just after section 5 in the same **run** command (If this section fails to clear a blocked port, use the appropriate "CLEAR" section step).

WARNINGS and CAUTIONS:

When one of this section's steps is executed, any user connected to the selected port on the HP-PB MUX, may lose data, be locked up, or may be disconnected.

Section 8 - HALT

This section is used to reset the HP-PB MUX card's firmware and hardware, while preserving the state of the RAM.

A `cmd_reset` is issued to the SE on the HP-PB MUX card under test. The MUX 68000 on-board processor is reset, the MUX firmware stops running, and the on-board ROM interpreter becomes active. Selftest will not be executed and microcode will not be downloaded. The state of HP-PB MUX memory will not be altered; therefore, the "DUMP" section (refer to Section 9) could be used to capture the memory contents. If this step is successful, the HP-PB MUX card is left in a state where it can accept other commands (If the HP-PB MUX card is not operable after this step, try the HP-PB MUX "CLEAR" section, referred to in Section 2).

WARNINGS and CAUTIONS:

HP-PB MUX, when this section is executed any user connected to any port on the may lose data, be locked up, or may be disconnected.

Section 9 - DUMP

This section provides the ability to display, or capture to a file, the contents of HP-PB MUX local RAM. The name of the `DUMP` file has the form: **DUMPxDyymmddhhmm** with **x** being the MUX number, **yy** the year, **mm** the month, **dd** the day, **hh** the hour, and **mm** the minutes. The `DUMP` file is in binary form, and can be listed in a hexadecimal format with the UNIX command **xd**.

Step 900 This step executes a dump of the entire MUX RAM. It can take a long time to execute (at least 15 minutes), and requires enough space on the disk to create the dump file (512 kb per dump file).

Step 901 This step executes a partial dump of part of the MUX RAM memory, which contains all the variables used by the firmware. It can take at least 15 seconds to execute, and creates a dump file of about 32 Kb.

Although microcode need not be loaded to use these steps, the HP-PB MUX card must be in a state ready to accept commands (this may require execution of the **HALT** section, prior to using the step).

The interpretation of the output from these steps must be done by an expert diagnostician, familiar with the HP-PB MUX card.

WARNINGS and CAUTIONS:

None - this section has no destructive effects on user data connected to the HP-PB MUX.

Error Messages

The following is a listing of the most representative error messages generated and displayed by PMUXDIAG.

5000 `*** DIAG INIT FAILED: CANNOT OPEN THE DEVICE DIRECTORY (MUXERR 5000)`

ACTION Check for the existence of and for user's capabilities on the /dev and /dev/diag directories.

5001 `*** DIAG INIT FAILED: CANNOT STAT THE DEVICE FILE (MUXERR 5001)`

5002 `*** DIAG INIT FAILED: INVALID DEVICE FILE:`

```
NAME TOO LONG (32 CHARACTERS MAX) OR
MUX OR PORT NUMBER OUT OF RANGE
CHECK THE /DEV/DIAG SPECIAL FILES (MUXERR 5002)
```

5003 `*** DIAG INIT FAILED: INVALID MUX NUMBER IN THE RUN COMMAND.`

```
CHECK THE /DEV/MUX AND /DEV/DIAG/MUX SPECIAL FILES (MUXERR 5003)
```

5004 `*** DIAG INIT FAILED: INVALID OPEN DEVICE IN THIS CONTEXT (MUXERR 5004)`

CAUSE PMUXDIAG is not the right program for this board. PMUXDIAG runs on 40299A, 40299B, J2092A, J2093A, J2094A, A1751-60001, J1703-60004, and J1703-60022 muxes only.

5005 `*** OPEN FAILED: DAR NOT INITIALIZED (MUXERR 5005)`

5006 `*** OPEN DEVICE FAILED: A DEVICE IS ALREADY OPEN (MUXERR 5006)`

5007 `*** OPEN DEVICE FAILED: DAR INTERNAL ERROR (MUXERR 5007)`

```
INVALID MUX CARD NUMBER.  CHECK DEVICE FILES.
```

5008		`*** OPEN DEVICE FAILED: ILLEGAL DEVICE (MUXERR 5008)`
5009		`*** OPEN DEVICE FAILED: DAR DEVICE TABLE ENTRY INCORRECT (MUXERR 5009)`
		`CHECK THE /DEV/DIAG DIAGNOSTIC SPECIAL FILE`
5010		`*** OPEN DEVICE FAILED: SYSTEM CALL FAILED (MUXERR 5010)`
5011		`*** CLOSE FAILED: DAR NOT INITIALIZED (MUXERR 5011)`
5012		`*** CLOSE FAILED: NO DEVICE OPEN (MUXERR 5012)`
5013		`*** CLOSE FAILED: INVALID FILE ID (MUXERR 5013)`
5014		`*** CLOSE FAILED: SYSTEM CALL FAILED (MUXERR 5014)`
5015		`*** CLEAR FAILED: MUX INITIALIZATION FAILED (MUXERR 5015)`
CAUSE		Suspected failing FRU is the mux.
5016		`*** CLEAR FAILED: INITIALIZATION ERROR FOR PORT ! (MUXERR 5016)`
CAUSE		Suspected failing FRU is the mux.
5017		`*** RESET FAILED: PARTIAL INITIALIZATION FAILED PORT ! (MUXERR 5017)`
ACTION		Try the clear section to reset the port. Be careful that it is a destructive section.
5018		`*** HALT FAILED: UNABLE TO STOP MUX ACTIVITY (MUXERR 5018)`

5019 `*** UNKNOWN DAR FAILURE RESULT (MUXERR 5019)`

5020 `*** MUX IS MARKED AS BROKEN (MUXERR 5020)`

5021 `*** RETURNED STATUS = ! (MUXERR 5021)`

5022 `*** INCORRECT TYPE OF MODULE: ! ; I EXPECTED: ! (MUXERR 5022)`

CAUSE PMUXDIAG is not the right program for this board. PMUXDIAG runs on 40299A, 40299B, J2092A, J2093A, J2094A, A1751-60001, J1703-60004, and J1703-60022 muxes only.

5023 `*** IODC CHECKSUM FAILED: ! (MUXERR 5023)`

CAUSE Bad IODC bytes. The mux ROMs may be corrupted. Suspected failing FRU is the mux.

5024 `*** INVALID IODC HVERSION NUMBER (MUXERR 5024)`

5025 `*** INVALID PARAMETER (MUXERR 5025)`

5026 `*** INTERNAL DIAGNOSTIC ERROR (MUXERR 5026)`

5027 `*** TEST FAILED. (MUXERR 5027)`

```
DATA = !
```

5028 `*** LOOPBACK FAILED BECAUSE THE MUX IS NOT LOADED (MUXERR 5028)`

```
RUN THE CLEAR SECTION FIRST (DESTRUCTIVE)
SECTION 2 STEP 199.
```

5029 *** I/O FAILURE OCCURRED TRYING TO DO LOOPBACK. (MUXERR 5029)

5030 *** LOOPBACK TEST FAILED. (MUXERR 5030)

FAILURE CODE = !

DATA = !

PORT = !

CAUSE Step in Section 4 have failed.

ACTION Ensure that Loopback hoods are fitted if required, and that peripheral cables are removed from the ADP, DDP or port under test before re running. If the test still fails, run tests starting at backplane and progressing to Panel. This will allow you to identify the failing unit.

5031 *** MUX CARD SELFTEST ! FAILED. (MUXERR 5031)

THE MUX CARD IS THE FAILING FRU.

ACTION If selftest number = 4, then replace the ROMs on the mux card. Otherwise, replace the mux card.

5032 *** CANNOT CREATE OR OPEN THE DUMP FILE (MUXERR 5032)

CHECK USER'S CAPABILITIES ON THE CURRENT DIRECTORY.

5033 *** DUMP MEMORY ADDRESS ERROR (MUXERR 5033)

5034 *** CANNOT WRITE TO THE DUMP FILE (MUXERR 5034)

CHECK USER'S CAPABILITIES ON THE CURRENT DIRECTORY.

5035 *** READ SYSTEM CALL FAILED (MUXERR 5035)

READER COMMENT SHEET

Systems Technology Division

Precision Architecture RISC:
Support Tools Manual (Licensed Users)
Vol. 2, Device Adapters/Muxes

Manual Part Number 5960-3151 December 1992

A reader comment sheet helps us to improve the readability and accuracy of the document. It is also a vehicle for recommending enhancements to the product or manual. Please use it to suggest improvements.

SERIOUS ERRORS, such as technical inaccuracies that may render a program or a hardware device inoperative, should be reported to your HP Response Center or directly to a Support Engineer. An engineer will enter the problem on HP's STARS (Software Tracking and Reporting System). This will ensure that critical and serious problems receive appropriate attention as soon as possible.

Editorial suggestions (please include page numbers): ______________________________

____

____

____

Recommended improvements (attach additional information, if needed): ____________________

____

____

____

Name: ________________________________ **Date:** ______________

Job Title: ______________________________ **Phone:** ____________

Company: ____

Address: ____

____

☐ Check here if you would like a reply.

Hewlett-Packard has the right to use submitted suggestions without obligation, with all such ideas becoming property of Hewlett–Packard.

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070, CUPERTINO, CA

– POSTAGE WILL BE PAID BY ADDRESSEE –

Publications Manager – Hardware
Hewlett-Packard Company
Systems Technology Division
19483 Pruneridge Avenue
Mailstop 42UK
Cupertino, California 95014-9786

Fold and Tape