HP OpenVMS Industry Standard 64 Evaluation ReleaseVersion 8.1
New Features and Release Notes

HP OpenVMS Industry Standard 64 Evaluation Release Version 8.1 for Integrity Servers

This is a new manual.

HP OpenVMS Industry Standard 64
Evaluation Release Version 8.1
is for evaluation purposes only and
is not intended for use
in production environments.

Manufacturing Part Number: n.a.
December 2003

Legal Notice

The HP OpenVMS documentation set is available on CD-ROM.

Contents

Contents

Contents

Contents

Preface

The book provides information about the HP OpenVMS I64 Evaluation Release Version 8.1.

Intended Audience

Anyone who plans to build and install the OpenVMS I64 operating system, and develop applications needs to read this manual.

Document Structure

This guide contains the following chapters and appendixes:

Chapter 1, Creating and the OpenVMS I64 Evaluation Release Version 8.1 Environment – describes how to boot and installation OpenVMS I64 operating system.

Chapter 2, General Release Notes and New Features – describes the release notes and new features for the general user audience.

Chapter 3, Programming Release Notes – describes release notes and new features pertaining to the programming audience.

Appendix A, Installation Script – provides a sample installation script for the OpenVMS I64 Evaluation Release Version 8.1 operating system.

Appendix B, Linker Maps – contains seven sample Linker maps.

Appendix C, Extensible Firmware Interface (EFI) Boot Manager– describes the EFI interface commands.

Appendix D, Management Processor Console Commands – contains the description of the management processor commands.

Appendix E, BMC Console Commands– contains a description of the BMC console commands.

Appendix F, Alpha Tools – provides a description of the Alpha tools you can to to perform tasks on your Alpha system that help you set up and manager your Integrity Server.

Related Documents

In addition to this manual, the following manuals are included in the SDK kit:

- *HP OpenVMS Calling Standard*
- *HP OpenVMS MACRO Compiler Porting and User's Guide*
- *HP Guide to Porting Applications from OpenVMS Alpha to OpenVMS Industry Standard 64 Integrity Servers*
- *HP OpenVMS System Services Reference Manual: A—GETUAI*
- *HP OpenVMS Sysetm Services Reference Manual: GETUTC—Z*

For additional information about HP OpenVMS products and services, see the following World Wide Web address:

`http://www.hp.com/products/openvms`

Reader's Comments

HP welcomes your comments on this manual.

Please send comments to either of the following addresses:

Internet: openvmsdoc@hp.com

Postal Mail:
Hewlett-Packard Company
OSSG Documentation Group
ZKO3-4/U08
110 Spit Brook Road
Nashua, NH 03062-2698

How to Order Additional Documentation

For information about how to order additional documentation, visit the following World Wide Web address :

`http://www.hp.com/go/openvms/doc/order/`

Conventions

The following conventions may be used in this manual:

Convention	Meaning
Ctrl/x	A sequence such as Ctrl/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 x	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key (x) or a pointing device button.
Return	In examples, a key name in bold indicates that you press that key.
…	A horizontal ellipsis in examples indicates one of the following possibilities: – Additional optional arguments in a statement have been omitted. – The preceding item or items can be repeated one or more times. – Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that you must enclose choices in parentheses if you specify more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.

Convention	Meaning
|	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are optional; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold type	Bold type represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason. In this manual, bold type also represents user input in interactive displays and code examples.
italic type	Italic type indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where (*dd*) represents the predefined par code for the device type).
UPPERCASE TYPE	Uppercase type indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Example`	This typeface indicates code examples, command examples, and interactive screen displays. In text, this type also identifies URLs, UNIX command and pathnames, PC-based commands and folders, and certain elements of the C programming language.
–	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1 Creating the OpenVMS Industry Standard 64 Evaluation Release Version 8.1 Environment

This manual describes how to set up your hardware environment and boot and install the OpenVMS Industry Standard 64 Evaluation Release Version 8.1 operating system. It also contains new features and release notes. This manual assumes that you are using the SDK kit for this release.

This chapter describes the process of building the OpenVMS operating system media and installing and booting it on an I64 system.

IMPORTANT HP OpenVMS Industry Standard 64 Evaluation Release Version 8.1 is for evaluation purposes only and is not intended for use in production environments.

To set up your environment, you need to do the following:

1. Unpack and hook up your Itanium® system.
2. Power up the machine and set up the Management Console (MP). See "Setting the Console Device" on page 17.
3. Log in on the MP Console using the serial interface.
4. Determine if you have the correct version of the firmware by typing the SYSREV console command. The current minimum firmware versions are:

   System Firmware: 2.20
   MP Console Firmware: E2.20

   If you do not have the correct version of firmware, proceed to step 5; otherwise, go to step 9.
5. To download the correct version of firmware, go to the following web site:

   `http://www.hp.com/support/itaniumservers`
6. Click on the appropriate server in the list, click on the link ,"download the drivers and software", then click on the link, "Cross operating system (BIOS, Firmware, etc.)".
7. Download the appropriate firmware onto a CD-ROM.
8. Install the console firmware.
9. Set the MP network address (TCP/IP) (optional). See "Configuring the LAN Console (Optional)" on page 23. Make sure that the Ethernet cable is plugged in.
10. Insert the OpenVMS DVD (or disk).
11. Boot the I64 system. See "Booting OpenVMS from the MP Console" on page 24.
12. Install the OpenVMS I64 Evaluation Release Version 8.1 operating system. See Appendix A, "Installation Script," on page 109, for a sample OpenVMS I64 installation script.
13. Remove the DVD and powercycle your system.

Tested Configuration for Version 8.1

Tables 1-1 and 1-2 list the recommended hardware configurations for the OpenVMS Industry Standard 64 Evaluation Release Version 8.1.

Table 1-1 HP rx2600 Recommended System Configuration

Part #	Quantity	Description
A6870B	1	HP rx2600 1.3GHz CPU server solution
A9872A	0-1	HP rx2600 1.3GHz with 3MB cache
A9910A	1	4G DDR memory quad
A6829A	1	Dual-channel Ultra 160 SCSI adapter card
A9919A	1	Read-only Optical Drive (DVD+R/CD+R)
A6825A	1	Single Port GigE-TX (gigabit copper) **(OR)**
A6847A	1	Single Port GigE-SX (gigabit fiber)
AB232A	1	PCI-X 1-port FCA2404 2GB FC
A9896A	1-4	36GB 15k rpm Ultra3 SCSI disk drive

OpenVMS Engineering is using this configuration for the development and qualification systems. This systems configured with these options, are recommended for use with the OpenVMS I64 Evaluation Release Version 8.1.

Standard system features include:

- Dual-channel Ultra320 SCSI controller, two internal disks on one channel, one internal disk on a second channel. External Ultra320 SCSI port. 10/100/1000Base-TX LAN (auto speed sensing, RJ-45 connector)
- 1.3 GHz CPUs are recommended.
- One more CPU can be added for SMP (total of two CPUs maximum).
- The amount of memory required may vary, but the minimum should be 1 GB with a maximum of 24 GB.
- The embedded GigE-TX LAN will run Tbase10/100/1000 speeds.
- The 2 GB FC adapter from Emulex (AB232A PCI-X 1-port FCA2404 2 GB fibre channel adapter) is also known as KGPSA-EA (LP9802) on AlphaServers.

Table 1-2 HP rx4640 Recommended System Configuration

Part #	Quantity	Description
A6961A	1	HP rx4640 1.3 GHz CPU server solution
A7159A	0-3	HP rx4640 1.3 GHz Itanium 2 CPU with 3MB cache
A6967A	1-4	1GB memory quad
A6829A	1	Dual-channel Ultra 160 SCSI adapter card
A7163A	1	Read-only Optical Drive (DVD+R/CD+R)
A6825A	1	Single Port GigE-TX adapter card
AB232A	1	PCI-X 1-port FCA2404 2 GB FC
A9896A	1-4	36GB 15k Hot Plug Ultra 320 SCSI

This system, configured with these options, are recommended for use with the OpenVMS I64 Evaluation Release Version 8.1.

Standard system features include:

- One processor and one power supply.
- A dual-channel Ultra160 SCSI controller and a single-port GigE-TX LAN, included as core I/O. The rx4640 does not include memory or hard disk drives. This server is optimized for rack environments and includes a rack kit.
- 1.3 GHz CPUs are recommended .
- Up to three more CPUs can be added for SMP (maximum of 4 CPUs).
- The amount of memory required may vary, but the minimum should be 1 GB with a maximum of 8 GB.
- The embeded GigE-TX LAN will run Tbase 10/100/1000 speeds.
- The 2 GB FC adapter from Emulex (AB232A PCI-X 1-port FCA2404 2GB Fibre Channel adapter) is also known as the KGPSA-EA (LP9802) on AlphaServers.

Management Processor Console

The management processor is an independent support system for the server. It provides a way for you to connect to a server and perform administration or monitoring tasks for the server hardware.

The management processor controls the following:

- Power
- Reset
- Transfer of control (TOC) capabilities
- Console access

- Display and recording of system events
- Display of detailed information about the various internal subsystems.

Accessing the Management Processor

You can connect to the management processor using the following methods:

- The local RS-232C port using a local terminal.
- The remote RS-232C port using external modem (dial-up) access, if remote modem access is configured.
- The management processor LAN port using Web Console or Telnet, if login access through the management processor LAN is enabled.

To interact with the management processor, perform the following steps:

1. Log in using your management processor user account name and password. Note, if the management processor is not displaying the MP MAIN MENU, use Ctrl/B to access the MP MAIN MENU and the management processor (MP) prompt.
2. Use the management processor menus and commands as needed. A list of available commands can be displayed by using the management processor help function (In the MP MAIN MENU, enter HE followed by LI at the MP HELP: prompt.). Log out using the X command (In the MP MAIN MENU, enter X at the MP> prompt.) when done.

Management Processor Command Interface

Use the management processor menus and commands as needed. The login screen, which includes the MAIN MENU, is shown below. Main menu commands (CO, VFP, CM, CL, CSP, SE, SL, HE, and X) can be entered after the MP prompt. Commands not displayed in the MP MAIN MENU can be accessed in command mode by first using the CM command at the MP prompt. (A list of available commands can be displayed by using the management processor help function. Display the list of commands as follows: in the MP MAIN MENU, enter HE after the MP> prompt, then enter LI after the MP HELP: prompt.) You can return to the MP MAIN MENU by typing Ctrl/B.

```
                              MP Welcome Screen


MP Login: Admin
MP password: ~
                           Hewlett-Packard Management Processor
     (C) Copyright Hewlett-Packard Development Company, L.P. 1999-2003. All rights
                                      reserved
                           System Name: xxxxxxxxx



          MP MAIN MENU:
             CO:Console
             VFP:Virtual Front Panel
             CM:Command Menu
             CL:Console Log
```

```
SL:Show Event Logs
CSP:Connect to Service Processor
SE:Create OS Session
HE:Main Menu Help
X:Exit Connection MP>
```

See Appendix D, “Management Processor Commands,” on page 153, for information about MP commands.

Setting the Console Device

The following steps set the Management Processor Console as your console device.

1. **Power on the system and log in.**

   To log into the MP console, you may need to press carriage-return a couple of times. You are then prompted to enter the username and password. By default, they are set to "Admin". If you are not prompted to login, you have old firmware and need to see the section on updating the MP firmware.

   You should initially hook up to the console line on the management processor.

```
******************************************************************
 Only default users are configured.
 Use one of the following user/password pairs to login:

 Admin/Admin
 Oper/Oper
******************************************************************

MP login: Admin
MP password: *****


                    Hewlett-Packard Management Processor

   (c) Copyright Hewlett-Packard Development Company, L.P. 1999-2003.  All Rights Reserved.

                          MP Host Name: myhost

                             Revision E.02.22

******************************************************************
                       MP ACCESS IS NOT SECURE
```

```
 Default MP users are currently configured and remote access is enabled.
 Modify default users passwords or delete default users (see UC command)
                                  OR
           Disable all types of remote access (see SA command)
*****************************************************************

   MP MAIN MENU:

         CO: Console
         VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Log
         SL: Show Event Logs
         CSP: Connect to Service Processor
         SE: Enter OS Session
         HE: Main Help Menu
         X: Exit Connection

MP Host Name: myhost
```

2. Type CM at the MP> prompt to enter command mode.

```
MP> cm

        (Use Ctrl-B to return to MP main menu.)
MP Host Name: myhost
```

3. Type pc -on -nc to power on the system with no confirmation.

```
MP:CM> pc -on -nc
PC -on -nc

System will be powered on.

   -> System is being powered on.

-> Command successful.

MP Host Name: myhost
```

4. Exit command mode by typing Ctrl/B.

```
MP:CM> [Crtl/B]
   MP MAIN MENU:

         CO: Console
         VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Log
         SL: Show Event Logs
         CSP: Connect to Service Processor
         SE: Enter OS Session
         HE: Main Help Menu
          X: Exit Connection

MP Host Name: myhost
```

5. Type co to return to console mode.

```
MP> co
```

As the system starts up, you see boot messages followed by the EFI Boot Manager screen.

```
EFI Boot Manager ver 1.10 [14.61]  Firmware ver 2.20 [4331]

Please select a boot option

    EFI Shell [Built-in]
    Boot Option Maintenance Menu
    System Configuration Menu

    Use ^ and v to change option(s). Use Enter to select an option.
```

From the Boot Manager , you must select the serial console as the sole input, output, and error device from the system.

6. From the EFI Boot Manager, select the Boot Option Maintenance Menu using either the letter v or the caret (^) or an up arrow or down arrow to move the cursor.

```
EFI Boot Maintenance Manager ver 1.10 [14.61]

Main Menu. Select an Operation

        Boot from a File
        Add a Boot Option
        Delete Boot Option(s)
```

```
        Change Boot Order

        Manage BootNext setting
        Set Auto Boot TimeOut

        Select Active Console Output Devices
        Select Active Console Input Devices
        Select Active Standard Error Devices

        Cold Reset
        Exit



    Timeout-->[10] sec SystemGuid-->[C198BA79-478A-11D7-9C22-6033AC66036B]
    SerialNumber-->[US30464638          ]
```

7. Select Active Console Output Devices, and move to the line labeled:

```
Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(Vt100)
```

8. Press Enter to toggle the selection, so that an asterisk appears to its left. Configure the system so that only that line is active; no other lines should have asterisks. Pressing the space bar changes the state.

9. Select "Save Settings to NVRAM".

10. Repeat Steps 7-9 to select the Active Console Input Devices and Active Standard Error Devices and save the settings.

NOTE For this release, the console devices must point to a serial line console.

For this example, we are changing only the console output device. When you select the output device, you receive a display something like the one below:

```
EFI Boot Maintenance Manager ver 1.10 [14.61]


Select the Console Output Device(s)


      Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(PcAnsi)
      Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(Vt100)
      Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(Vt100+)
      Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(VtUtf8)
      Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(PcAnsi)
    * Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(Vt100)
```

```
      Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(Vt100+)
      Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(VtUtf8)
      Acpi(HWP0002,700)/Pci(2|0)
    Save Settings to NVRAM
    Exit
```

11. Finally, perform a cold reset of the system. (If you changed the console from MP to the BMC or vice versa, you may need to move the serial cable to the appropriate connector on the system. See Appendix E, "BMC Console Commands," on page 163, for a list of BMC system console commands.)

Input, Output, and Error Console Device Restrictions

Currently, the OpenVMS operating system has two console restrictions for the MP console (The BMC console has the same restrictions.):

- The input, output, and error devices for these must point to a serial line console.
- They must all be the same serial device.

If these conditions are not met, you receive a warning out to the VMS_LOADER, and you may see other errors in later parts of the boot. Additionally, you may lose output that you would normally expect to see when booting.

Powering the System On/Off from the MP Console

This section describes the steps for powering the system on from the MP console.

1. Log into the MP console.

2. Switch to command mode by typing CM at the MP> prompt.

```
                          .
                          .
                          .
MP> cm

        (Use Ctrl-B to return to MP main menu.)


MP Host Name: myhost
MP:CM>
```

3. Turn on the power by entering the command, pc -on -nc, and then Ctrl/B to get out of command menu mode.

```
MP:CM> pc -on -nc
PC -on -nc
System will be powered on.


   -> System is being powered on.
```

```
-> Command successful.

MP Host Name: myhost
MP:CM> [Ctrl/B]

   MP MAIN MENU:
         CO: Console
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Log
         SL: Show Event Logs
        CSP: Connect to Service Processor
         SE: Enter OS Session
         HE: Main Help Menu
          X: Exit Connection

MP Host Name: myhost
MP>
```

4. To power off the system, return to the command menu and enter the command pc -off-nc.

```
MP> cm

(Use Ctrl/B to return to MP main menu.)

MP Host Name: myhost
MP:CM> pc -off -nc

PC -off -nc

System will be powered off.
```

You must shut down the OS manually before this command is executed. Failure to do this can cause problems when the OS is restarted.

```
   -> System is being powered off.

-> Command successful.
```

```
MP Host Name: myhost
MP:CM> exit


   MP MAIN MENU:


         CO: Console
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Log
         SL: Show Event Logs
        CSP: Connect to Service Processor
         SE: Enter OS Session
         HE: Main Help Menu
          X: Exit Connection


MP Host Name: myhost
MP>
```

Configuring the LAN Console (Optional)

You can plug a DECserver directly into the ‘pigtail” serial connector for remote console access. Alternatively, if you have a second TCP/IP address for the system, you can access the console by way of TCP/IP.

To configure the LAN console, press Ctrl/B to get to the MP> prompt, CM to get to the command menu, then LC to configure the LAN.

```
MP> CM
MP:CM> LC


At each prompt you may type DEFAULT to set default configuration or Q to Quit


                      Current LAN configuration:
                      -  -  MAC Address  :  address
                      I  -  IP address  : nn.nn.nn.nnn
                      H  - MP Host Name  :  hostname
                      S  - Subnet Mask  :  nnn.nnn.nnn.n
                      G  - Gateway Address  :  nn.nn.nn.nnn
                      L  - Link State  :  10BaseT
                      W  - Web Console Port Number  :  nnnn
Enter parameters(s) to change, A to modify All, or [Q] to Quit:  A
```

Follow the prompts to enter your IP address, hostname, and gateway. After applying the changes, you can disconnect your PC or terminal and begin accessing the console over TCP/IP using the telnet command.

Booting OpenVMS from the MP Console

The following steps describe how to boot OpenVMS from the MP console:

1. Log into the MP console.

2. Switch to command mode by typing CM at the MP> prompt.

```
MP> cm

        (Use Ctrl-B to return to MP main menu.)

MP Host Name: myhost
MP:CM>
```

3. Turn on the power by entering the command pc -on -nc and then Crtl/B to get out of command menu mode.

```
MP:CM> pc -on -nc

PC -on -nc
System will be powered on.
   -> System is being powered on.
-> Command successful.
MP Host Name: myhost
MP:CM> [Ctrl/B]
   MP MAIN MENU:

         CO: Console
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Log
         SL: Show Event Logs
        CSP: Connect to Service Processor
         SE: Enter OS Session
         HE: Main Help Menu
          X: Exit Connection

MP Host Name: myhost
MP> CO [ENTER]
```

4. Exit MP command mode by typing the CO command and wait for the EFI command menu. (Note that some systems may have been set to time out after a small number of seconds.)

```
EFI Boot Manager ver 1.10 [14.61]  Firmware ver 2.20 [4331]

Please select a boot option

    EFI Shell [Built-in]

    Boot Option Maintenance Menu

    System Configuration Menu

    Use ^ and v to change option(s). Use Enter to select an option
```

5. Use the letter v or the caret (^) to select the "EFI Shell [Built-in]". You then receive the Shell> prompt:

```
Loading.: EFI Shell [Built-in]

EFI Shell version 1.10 [14.61]

Device mapping table

  fs0  : Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)/HD(Part1,SigC1CEDDA4)

  blk0 : Acpi(HWP0002,0)/Pci(2|0)/Ata(Primary,Master)

  blk1 : Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)

  blk2 : Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)/HD(Part1,SigC1CEDDA4)

Shell>
```

6. Enter FS0: or the file structured device that has OpenVMS on it.

```
Shell> fs0: [ENTER]

fs0:\>
```

7. Now change the directory containing the OpenVMS initial boot loader by entering cd \efi\boot.

```
fs0:\> cd \efi\boot

fs0:\efi\boot>
```

8. Invoke the initial loader by entering bootia64.efi -fl 0,2. The -fl is one way to specify boot time flags just like on the Alpha system with the SRM console.

```
fs0:\efi\boot> bootia64.efi -fl 0,2
```

The OpenVMS I64 installation script starts at this point. This script is similar to the OpenVMS Alpha installation script. See Appendix A, “Installation Script,” on page 109, for a sample OpenVMS I64 installation. For a complete description of an installation, see the *HP OpenVMS Alpha Version 7.3-2 Upgrade and Installation Manual* at the following web site:

```
http://www.hp.com/go/openvms/doc
```

EFI Postinstallation Options

Power on the Integrity server.

NOTE On the rx2600, you may see a warning that the event log is full. You can safely continue by pressing the spacebar.

If you want to clear the event log, press Ctrl/B to drop to the MP> prompt, then select SL and use the "C" option to clear the log.

At the Boot Manager menu, select "EFI Shell [Built-in]".

Perform the optional configuration steps as described in the following sections. When you are finished, boot OpenVMS by typing:

```
Shell> fs0:
fs0:\> cd\efi\vms
fs0:\efi\vms> vms_loader.efi
```

You see a series of messages, and eventually, you receive the system login prompt.

Setting OpenVMS Boot Flags (Optional)

To set the OpenVMS boot flags, at the Shell> prompt, type:

```
SET VMS_FLAGS "0,0"
```

If desired, you may use any of the standard VMS boot flags:

```
SET VMS_FLAGS 0,1 Enable SYSBOOT to change system parameters
SET VMS_FLAGS 0,2 -- Load XDELTA
SET VMS_FLAGS 0,4 -- Take the initial EXEC_INIT breakpoint
SET VMS_FLAGS 0,20000 -- Print debug messages on boot
SET VMS_FLAGS 0,30000 -- Print even more debug messages on boot
```

Automatic Boot of OpenVMS (optional)

To boot automatically, add a new entry on the EFI Boot Manager menu, so that you do not need to go to the EFI shell every time you boot. To add a new entry on the VMS Boot Menu, enter the EFI shell, and type:

```
Shell> bcfg boot add 1 fs0:\efi\vms\vms_loader.efi "hp OpenVMS I64"
```

In addition, you can add an EFI boot menu option by using the EFI menu interface:

1. Select “Boot Options Maintenance Menu”.
2. Select “Add a Boot Option”.
3. Select the boot device and boot file.

The system prompts you for an optional boot flag, but none is required because the menu selection uses the VMS_FLAGS environment variable. You are also prompted for a menu name, which you should begin with OpenVMS.

NOTE All EFI boot options embed the disk Globally Unique ID (GUID). Therefore, if you re-install OpenVMS or restore a system disk from an image backup, you must first delete the old boot options, and then add a new boot option.

After booting and running AUTOGEN, you will receive several messages at DECwindows startup. See “HP DECwindows Motif for OpenVMS Release Notes” on page 31, for information on these messages and how to avoid them.

Ctrl/H and the Delete Key

Unlike the OpenVMS operating system, which uses the character 0X7F DEL/RUBOUT, the MP and BMC consoles and the EFI environment use Ctrl/H. (See Appendix E, “BMC Console Commands,” on page 163, for a list of BMC system console commands.) If you press the Delete key on a VTxxx terminal or the key you have mapped to send 0X7F in your terminal emulator, the character is not deleted.

A new SYSGEN parameter, TTY_DEFCHAR3, and a new SET TERMINAL command provide the means by which you can tell the OpenVMS terminal driver to re-map Ctrl/H to DEL. If you want the re-mapping to be system wide for all terminals, you need to OR 0x10 into the current value stored in TTY_DEFCHAR3. If you want to re-map a single terminal, issue the following command:

```
$ set terminal/backspace=delete
```

Doing this allows you to re-map Ctrl/H to DEL, the driver does not perform the re-mapping operation if the terminal is in one of the following states:

- Terminal attribute is set to PASSALL
- Terminal attribute is set to PASTHRU
- IO$_READALL
- IO$_READPBLK
- Entering Ctrl/V tells the driver to pass the next character and skip the re-map check.

OpenVMS Postinstallation Configuration

The following sections describe tasks you can perform after you have installed the OpenVMS I64 operating system.

Modifying SYSGEN Parameters (Mandatory)

OpenVMS I64 Evaluation Release Version 8.1 requires the VMSD2 SYSGEN parameter to be set to 1. Add the following line, VMSD2=1, to SYS$SYSTEM:MODPARAMS.DAT, then run AUTOGEN as follows:

```
$ @sys$update:autogen getdata setparams
```

This requirement will be removed in a future release.

Creating a Batch Queue

The default batch queue runs on nodename that you specify. You must recreate the queue using the new nodename. Perform the following steps:

1. `Mynode> start/queue/manager/new`
2. `init/queue sys$batch`
3. `Mynode> init/queue sys$batch/batch/job=100/on=`*`nodename`*`::/start`
4. `Mynode> show queue`

   ```
   Batch queue SYS$BATCH, idle, on
   ```

Editing System Startup Files

If desired, you may edit the system startup files:

```
$ EDIT SYS$STARTUP:SYLOGICALS.COM
$ EDIT SYS$STARTUP:SYSTARUP_VMS.COM
```

We recommend the following additions:

```
$ DEFINE/SYS/EXEC SYSUAF SYS$SYSTEM:SYSUAF.DAT
$ DEFINE/SYS/EXEC RIGHTSLIST SYS$SYSTEM:RIGHTSLIST.DAT
$ START/QUEUE SYS$BATCH
```

Formatting Error Logs

When formatting OpenVMS error logs, we recommend that you copy the error log file from the OpenVMS I64 system to an OpenVMS Alpha system. Then format the error log file on the OpenVMS Alpha system using the Error Log Report Formatter (ERF).

Configuing HP TCP/IP Services for OpenVMS on Version 8.1

TCP/IP can be selected for installation when the PCSI installation is run on the HP OpenVMS Industry Standard 64 Evaluation Version 8.1 disk. After installation, system or application managers can configure TCP/IP by entering:

```
@sys$manager:tcpip$config
```

For the TCP/IP release notes, if you have TCP/IP installed, you will find them in the directory, `SYS$HELP:TCPIP055.RELEASE_NOTES`. You can also extract them from the kit by using the following command:

```
$ product extract release_notes
```

Configuring HP DECnet for OpenVMS Phase IV on Version 8.1

To configure DECnet Phase IV, do the following:

- At the DCL prompt, type:

  ```
  $ @netconfig
  ```

- Enter your nodename and DECnet address when prompted. For example:

  ```
  DECnet node  : mynode
  DECnet address  :  12.345
  ```

- Start the network

If you want the network to start whenever you boot, add the command $@SYS$MANAGER:STARTNET to your SYSTARTUP_VMS.COM file. At this point, you should be able to SET HOST and copy files to and from your system.

Configuring HP DECdfs for OpenVMS on Version 8.1

Unlike Version 8.0, DECdfs will not be preinstalled on the OpenVMS Industry Standard 64 Evaluation Version 8.1 disk. Instead, DECdfs will be distrbuted as a VMSINSTAL kit. After DECnet is configured and started, you can install DECdfs started with a command similar to the following:

```
$ @SYS$UPDATE:VMSINSTAL HP-I64VMS-DECDFS_03023 kit-location
```

Due to the lack of a security server, you must answer NO to the following question:

```
Do you want to run the IVP after the installation [YES}?
```

Also, because of the lack of a security server, DECnet proxy files (NETPROXY.DAT and NET$PROXY.DAT) cannot be created under Version 8.1. These files must be created (with the appropriate proxy information for the Itanium node) on an Alpha node and then copied to the Itanium node.

Recovering from Hangs or Crashes

If your system hangs and you want to force a crash, from the console enter `Ctrl/P`. The method of forcing a crash dump varies depending on whether XDELTA was loaded or not.

If XDELTA was loaded, entering a `Ctrl/P` results in the system entering XDELTA. The system will display the instruction pointer and current instruction. A crash can be forced from XDELTA by entering `;C` as in the following example:

```
$

Console Brk at 8068AD40

8068AD40!            add         r16 = r24, r16 ;;  (New IPL = 3)

;C
```

If XDELTA was not loaded, issuing a `Ctrl/P` will result in the system responding with the prompt "Crash? (Y/N)". Typing a "Y" causes the sytsem to crash. Typing any other character results in the system continuing.

HP DECnet-Plus for OpenVMS Phase V Release Notes

This section discusses operational notes for HP DECnet-Plus for OpenVMS Industry Standard 64 for Integrity Servers Version I8.1. Most, if not all, of these operational notes are specific to this release and are not expected to impact future releases.

DECnet Release Notes Incorrect Concerning Cluster Alias

The DECnet-Plus Release Notes manual shipping with the DECnet-Plus I8.1 kit incorrectly states that Cluster Alias was ported, but not tested. This is incorrect. Cluster Alias is fully supported on OpenVMS I64 systems.

DECnet Over TCP/IP and OSI Over TCP/IP Status

The DECnet over TCP/IP and OSI over TCP/IP code have been ported and tested with the PWIP driver provided by the TCP/IP Engineering group. However, the PWIP driver has not been thoroughly tested by the TCP/IP Engineering group.

DECdns Server Not Supported

HP DECnet-Plus for OpenVMS Industry Standard 64 for Integrity Servers Version I8.1 does not support DECdns server operations. The DECdns client is supported.

MOP Operations Not Supported

HP DECnet-Plus for OpenVMS Industry Standard 64 for Integrity Servers Version I8.1 does not support MOP operations.

X.25 Data Links Not Supported

The HP X.25 for OpenVMS Alpha software is in the process of being ported and is not yet supported. Therefore, HP DECnet-Plus for OpenVMS Industry Standard 64 for Integrity Servers Version I8.1 does not support X.25 data links.

No Support for Creating DECnet Proxy Files

The Security Server is not yet available. Therefore, the DECnet proxy files, NETPROXY.DAT and NET$PROXY.DAT, cannot be created on an OpenVMS I64 Evaluation Release Version 8.1 system. Instead, create the files on an Alpha node and copy the files to the Version 8.1 system.

Supported Devices

HP DECnet-Plus for OpenVMS Industry Standard 64 Integrity Servers Version I8.1 supports two of the three Ethernet devices found in the rx2600 processor: the 10/100 Base-TX Ethernet LAN (device name EIA) and the 10/100/1000 Base-TX Ethernet LAN (device name EWA). The 10/100 Base-TX management LAN is not supported. Therefore, the only data link module supported is the CSMA-CD module.

HP DECdfs T2.3-3 Does Not Support DECnet Phase V

If DECnet-Plus is installed on your system, an attempt to install DECdfs T2.3-3 will fail. This problem will be fixed in a future release of OpenVMS.

HP DECwindows Motif for OpenVMS Release Notes

The following sections list HP DECwindows Motif for OpenVMS T1.4-1 release notes. DECwindows Motif T1.4-1 is installed using PCSI for OpenVMS I64 Evaluation Release Version 8.1 and DECwindows T1.4-1.

The DECwindows release notes are available at SYS$COMMON:[SYSHLP]DECW$MOTIFT014.RELEASE_NOTES.

Supported Keyboard

The keyboard supported for DECwindows on an OpenVMS I64 system is the LK463. Other keyboards may work, but are not supported.

Increase the Value of GH_RES_CODE

HP recommends that you increase the value of the GH_RES_CODE system parameter to a minimum value of 4096. This prevents the following messages from displaying at startup:

```
$ @SYS$MANAGER:DECW$STARTUP

%INSTALL-I_NONRES, image installed ignoring '/RESIDENT' DISK$IA64XA0W:
<SYS0.SYSCOMMON.SYSLIB>DECW$XLIBSHR.EXE
-INSTALL-E-NOGHREG, insufficient memory in the code or data granularity hint region
%INSTALL-I-NONRES, image installed ignoring '/RESIDENT" DISK$IA64XA0W:
<SYS0.SYSCOMMON.SYSLIB>DECW$XMLIBSHR12.EXE
-INSTALL-E-NOGHREG, insufficient memory in the code or data granularity hint region
%INSTALL-I-NONRES, image installed ignoring '/RESIDENT' DISK$IA64XA0W:
<SYS0.SYSCOMMON.SYSLIB>DECW$DXMLIBSHR12.EXE
-INSTALL-E-NOGHREG, insufficient memory in the code or data granularity hint region
```

NOTE Because GH_RES_CODE is not a dynamic SYSGEN parameter, you must reboot the system to apply the updated value.

Traditional Desktop is the Default for DECwindows Motif T1.4-1

Although the PCSI installation of the DECwindows Motif product indicates that the New Desktop is set as the default desktop, for Version T1.4-1 of DECwindows Motif, the default desktop is the Traditional Desktop.

To change from the Traditional Desktop to the New Desktop, edit the file SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM, change false to true, then restart DECwindows:

```
$ decw$startup_new_desktop == "false"

$ decw$startup_new_desktop == "true"

$ @sys$manager:decw$startup restart
```

After you modify DECW$PRIVATE_APPS_SETUP.COM and restart DECwindows, the next time you start a desktop, the desktop that appears should be the New Desktop.

New Desktop Password Field Lacks Backspace and Ctrl/U Editing Capabilities

There is a restriction with logging into the New Desktop for DECwindows Motif T1.4-1. When you type characters into the password field of the login dialog, you must be careful to type correctly. The ability to backspace within the password field is missing for DECwindows Motif T1.4-1. Also, the Ctrl/U editing functionality does not work for the password field. In previous versions of DECwindows, the Ctrl/U key sequence would remove all password characters typed, to allow the user to start again with password text entry.

The backspace and Crtl/U functionality for the New Desktop password field will be restored in the next release of DECwindows.

Graphics Card Support

The only graphics card currently supported on OpenVMS I64 systems is the embedded Radeon 7000 PCI card.

Hardware Acceleration Not Supported on Radeon 7000

For this release of DECwindows Motif T1.4-1 for OpenVMS, hardware acceleration (direct memory access, DMA) and 3D graphics are not currently supported on the Radeon 7000 PCI card.

Problem Restarting DECwindows Server

Because of a timing problem on some systems, DECwindows cannot be restarted with the command @SYS$MANAGER:DECW$STARTUP RESTART. To work around the problem, restart DECwindows using one of two methods:

- Stop the DECW$SERVER_0 process, and then start DECwindows with @SYS$MANAGER:DECW$STARTUP. Because stopping the DECW$SERVER_0 process ends the DECwindows session, these commands should be executed from the serial console or from another system using either the SET HOST command or telnet.
- Edit the command procedure SYS$MANAGER:DECW$STARTSERVER.COM. Search for the following line:

  ```
  $ write x "$STOP/ID=''pid''"
  ```

  Insert the following line after it:

  ```
  $ write x "$WAIT 0:0:5"
  ```

  Using this method allows the @SYS$MANAGER:DECW$STARTUP RESTART command to work normally.

Harmless Mouse or Keyboard Messages During DECwindows Startup

To configure the OpenVMS I64 system as a DECwindows server, you must have a USB mouse, an appropriate USB keyboard, and a display monitor connected to it. When running with your I64 system as a DECwindows client only (without the server configuration), you may see messages during DECwindows startup, indicating a 15-second countdown that is either “Waiting for mouse...” or “Waiting for keyboard...”. These messages are harmless. If you have a USB mouse and an appropriate USB keyboard, you can plug those into prevent the countdown messages during DECwindows startup.

HP DCE RPC for OpenVMS Release Notes

The following sections describe HP DCE RPC release notes that apply to the OpenVMS I64 Evaluation Release Version 8.1.

Support for G_FLOAT and IEEE FLoating Point on I64 Systems

DCE RPC for OpenVMS now supports both G_FLOAT and IEEE floating point types. Two versions of DCE Shared library are being shipped with this operating system version. The SYS$SHARE:DCELIB_SHR.EXE supports G_FLOAT type and SYS$SHARE:DCE$LIB_SHR_IEE.EXE support the IEEE floating point type. RPC applications built for I64 systems can use IEEE_FLOAT floating point types.

The following are the details for using the IEEE_FLOAT floating point type in RPC applications developed on the C language:

1. Compile the RPC application programs using the CC compiler qualifier /FLOAT_IEEE_FLOAT.
2. Use the -CC_CMD CC/FLOAT=IEEE_FLOAT IDL qualifier while building the stubs.
3. Link the RPC application with DCE:DCE_IEE.OPT.

The following are the details for using the G_FLOAT floating point type in RPC applications developed on the C language:

1. Compile the RPC application programs using the CC Compiler Qualifier /FLOAT=G_FLOAT.
2. Link the RPC application with DCE:DCE.OPT.

Multiple RPC applications, each using different floating point types, can be run on a system.

Windows NT LAN Manager

Authenticated RPC over NTLM (Microsoft's NT LAN manager protocol) is not supported in this version of DCE RPC for OpenVMS.

DCE Application Development Kit

To obtain a subset of the DCE Application Development Kit for building your application, contact your support team.

The utc_multime Factor Argument Type

The input argument, factor, for the DTSS API routine utc_mulftime must be a G_FLOAT type.

RTI (Remote Task Invocation) RPC

This version of RPC does not support RTI RPC.

HP C for OpenVMS I64 Release Note

When installing HP C Version T7.0-2 for OpenVMS I64 systems, set the default directory to a writeable directory, not the DVD media directory, to allow the IVP to succeed. For example:

```
$ set default sys$manager
$ product install C/source=DQA0:[C_i64.kit]
```

Note, this also applies to the HP COBOL Version T2.8 installation.

2 General Release Notes and Features

This chapter contains features and release notes pertaining to this baselevel of the HP OpenVMS Industry Standard 64 Evaluation Release Version 8.1 operating system.

IMPORTANT HP OpenVMS Industry Standard 64 Evaluation Release Version 8.1 is for evaluation purposes only and is not intended for use in production environments.

Clustering on OpenVMS I64

With few exceptions, OpenVMS Cluster software provides the same features on OpenVMS I64 systems as it currently offers on OpenVMS Alpha and VAX systems.

Key OpenVMS Cluster features include:

- Fully shared, multiple-node read/write disk access
- Clusterwide file system
- Clusterwide batch/print queue subsystem
- Distributed lock manager
- Votes/quorum-based membership management
- Shared system disk
- Single security domain
- Single system management domain
- Rich, clusterwide API
- Mixed-architecture clusters
- Support for rolling upgrades
- Support for multiple interconnects
- Support for a maximum of four systems, which can be Alpha and I64 systems or only I64 systems.

  For this release, the OpenVMS Cluster systems with two to four I64 systems have been tested.
- Failover and load balancing
- Cluster network alias
- Disk servers only. Tape servers will be supported in a future release.

Disaster-tolerant capabilities with support for distances up to 500 miles (800 kilometers) using Disaster-Tolerant Cluster Services (DTCS) are not supported in this release. They will be supported in a future release.

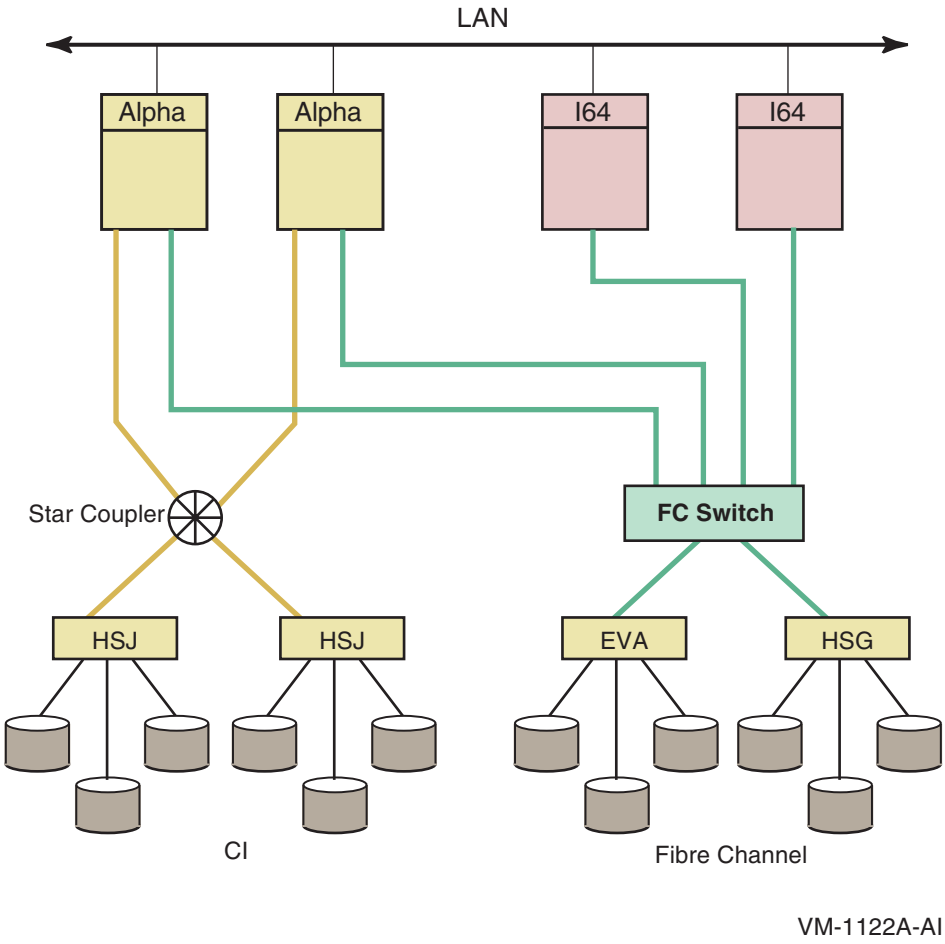
Mixed-Architecture Clusters

OpenVMS supports both OpenVMS Alpha systems and OpenVMS I64 systems in a mixed-architecture cluster. The OpenVMS Alpha versions supported in this configuration are Versions 7.3-1 and 7.3-2. Mixed-version support for all of these versions requires the installation of one or more remedial kits, as described in the *HP OpenVMS Alpha Version 7.3-2 Release Notes*. See the following web site for the HP OpenVMS Alpha Version 7.3-2 documentation set:

`http://www.hp.com/go/openvms/doc`

The following configuration shows an OpenVMS Cluster system to which two OpenVMS I64 systems have been added. A LAN interconnect is used for cluster communications for all systems in the cluster. In this configuration, the same Fibre Channel storage can be accessed by both OpenVMS Alpha and OpenVMS I64 systems at the same time. Note that the Itanium-based systems, directly connected to Fibre Channel disks, can be served data from the CI disks. In an OpenVMS mixed-architecture cluster, each architecture requires a minimum of one system disk. For this release, up to four systems are supported in a cluster. In a mixed-architecture cluster, this means you can include one, two, or three I64 systems with Alpha systems so that the total does not exceed four systems.

Figure 2-1 OpenVMS Cluster System with Alpha and OpenVMS I64 Systems



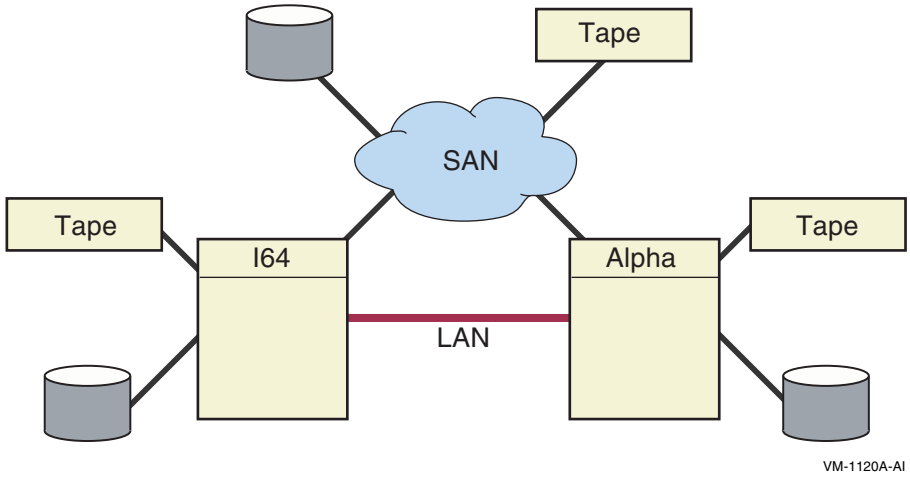
Storage in a Mixed-Architecture Cluster

This section describes the rules pertaining to storage, including system disks, in a mixed-architecture cluster consisting of OpenVMS I64 and OpenVMS Alpha systems.

These rules are in effect for this release. Additional functionality for tape serving on I64 systems is planned for a future release.

Figure 2-2 is a simplified version of a mixed-architecture cluster of OpenVMS I64 and OpenVMS Alpha systems with locally attached storage and a shared SAN. OpenVMS I64 systems can connect to the SAN using the LP9802 Fibre Channel host adapter (model number DS-KGPSA-EA).

Figure 2-2 Storage in a Mixed-Architecture OpenVMS Cluster



I64 systems in a mixed-architecture OpenVMS Cluster system:

- Must have their own system disk. Multiple I64 systems cannot share an I64 system disk. This restriction will be removed as soon as boot serving support (also known as satellite booting) for Fibre Channel storage becomes available.
- May run into problems while doing image backups to tape.
- Can use served Alpha disks but cannot use served Alpha tapes.
- Can use SAN disks and tapes.
- Can share the same SAN data disk with Alpha systems.
- Can serve disks to other cluster members, both I64 and Alpha systems.

Alpha systems in mixed-architecture OpenVMS Cluster systems:

- Must have an Alpha system disk, which can be shared with other clustered Alpha systems.
- Can use locally attached tapes and disks.
- Can serve disks to both I64 and Alpha systems, but can serve tapes only to other Alpha systems, not to I64 systems.
- Can use I64 served data disks.
- Can back up data on I64 served disks to Alpha tapes.

- Can use SAN disks and tapes.
- Can share the same SAN data disk with I64 systems.

OpenVMS Cluster Features Not Supported on OpenVMS I64 Systems

OpenVMS Cluster software supports three proprietary interconnects on Alpha systems that are not supported on OpenVMS I64 systems: DSSI (DIGITAL Systems Storage Interconnect), CI (Cluster Interconnect), and Memory Channel. Although DSSI and CI storage cannot be directly connected to OpenVMS I64 systems, data stored on CI and DSSI disks (connected to Alpha systems) can be served to OpenVMS I64 systems in the same cluster. In the configuration shown in Figure 2-1 on page 36, data stored on the CI disks can be served to the OpenVMS I64 systems.

Multihost shared storage on a SCSI interconnect, commonly known as SCSI clusters, is not supported. However, industry-standard Fibre Channel storage, on which storage area networks (SANs) are based, is supported.

Temporary OpenVMS I64 Cluster Restrictions

The following restrictions are limited to this release:

- I64 systems can use served Alpha disks, but cannot use served Alpha tapes. However, shared SAN storage on Fibre Channel is supported.
- No network booting (also known as satellite booting).
- Up to four I64 systems have been tested in an OpenVMS Cluster system.

HP Volume Shadowing for OpenVMS

HP Volume Shadowing is not supported on OpenVMS I64 Evaluation Release V8.1.

InfoServer Client Software

The InfoServer client software is not supported for the OpenVMS I64 Evaluation Release Version 8.1.

Availability of Native OpenVMS I64 Tools

The following tools run natively on the OpenVMS I64 Version 8.1 operating system. However, please note that these tools have not been fully tested.

- Linker
- Analyze/object

- Analyze/image, Librarian (both the DCL command and the services)
- Cross reference utility
- Command Definition Utility
- MACRO-32 compiler
- Message and Checksum utilities

Monitor Utility Release Notes

The Monitor utility has been ported as a native utility for OpenVMS I64 systems. Several enhancements were made to the utility during the port:

- Several classes of data (such as DISK) now utilize the entire screen height when displaying information.
- The SYSTEMs class now displays the number of "current" processes in Process States. Previously, current processes were grouped in the "Other" category.
- The format of the recorded data file has changed with this release. The format changes were made to improve the alignment of the recorded data. These changes do not allow pre-Version 8.0 nodes (Alpha) to read the new format of a recorded data file. There is however, a utility in SYS$EXAMPLES, which allows a new Version 8.0 format data file to be converted to the pre-Version 8.0 format. The utility is called SYS$EXAMPLES:MONITOR_CONVERT.C. An executable is also provided. The usage of this utility follows:

  ```
  $ mcr sys$examples:monitor_convert input-file output-file
  ```

- Pre-Version 8.0 nodes are unable to display live data (with either MONITOR/CLUSTER or MONITOR/NODE-node-name) from nodes running Version 8.1 or better.
- Various performance improvements were made to the MONITOR data collection routines to reduce the system overhead of using the Monitor utility.

The Monitor utility for OpenVMS I64 Version 8.1 has several issues:

- MONITOR/SUMMARY with multiple input files results in an access violation error. This will be fixed in a future release.
- Attempting to display MODE data from an old format INPUT file results in an access violation error. This will be fixed in a future release.
- Converting a data record by an OpenVMS I64 system, allows a pre-Version 8.0 Alpha system to play back the converted file. Converting recorded files with MODE data and an interval of 1 second results in a floating divide by zero if you are playing back this file on a pre-Version 8.0 node.

HP Secure Web Server for OpenVMS Release Notes

The following are known problems with the HP Secure Web Server Version 1.3 for OpenVMS:

- The mod_auth_openvms module is not operational. This module uses the OpenVMS sys$scan_intrusion system service, which is not available at this time.
- The Apache server processes do not handle signal delivery correctly. The following functions and operations result in an access violation and cause the servers to restart:
  - — Using the “KeepAlive on” directive in httpd.conf. Turn keep-alive off (“KeepAlive Off”)/
  - — $@APACHE$STARTUP [GRACEFUL|RESTART|NEW|FLUSH] causes the servers to crash. Use SHUTDOWN and START instead.

User Environment Test Package

For the OpenVMS I64 Evaluation Release Version 8.1, the User Environment Test Package (UETP) can be used with the following cautions:

- During the load phase, there are sporadic access violations in UETMEMY01. This does not terminate execution or really affect the validity of the run. UETP is still useable and produces valid results.
- The device phase currently does not complete execution due to an access violation.
- The DECnet phase runs fine. The cluster phase is still being tested. It appears to execute properly, but there are some concerns, and the output does not show other system names properly.

Support for RMS Journaling

The following RMS Journaling types are supported in Version 8.1:

- After-image (AI)
- Before-image (BI)

The third type of journaling, recovery unit (RU) journaling, is not supported on OpenVMS I64 Version 8.1.

DCL Commands That Currently Do Not Work

The following DCL commands currently do not work:

```
EDIT/TECO
SET SERVER
THREADCP
```

Other DCL Commands and Procedures

- **The SECURITY_SERVER** is not yet available. Due to the lack of a security server, DECnet proxy files (NETPROXY.DAT and NET$PROXY.DAT) cannot be created under Version 8.1 with the AUTHORIZE utility. These files must be created (with the appropriate proxy information for the I64 node) on an Alpha node and then copied to the I64 node.
- **SET PROCESS/DUMP[=NOW]** - SET PROCESS/DUMP can be used to modify the setting for the current process only. You cannot use SET PROCESS/DUMP to change the setting for another process or to initiate a dump of another process.
- **SHOW PROCESS/CONT** runs, but the pc and psl values are incorrect.
- **DCL Help** has not been revised for OpenVMS I64 tools. The version of DCL help that accompanies several OpenVMS I64 tools (Linker, Debugger, Librarian, Analyze) does not reflect the features and changes made for the I64 platform. As such, information is either not there supporting new features or is inaccurate.

  While the majority of DCL Help is still accurate, be sure to consult this release notes document to make sure you are receiving the most up-to-date information.

License Management Facility Note

For the OpenVMS I64 Evaluation Release Version 8.1, the License Management Facility (LMF) has been set to always return a “success” status to any calls to `SYS$GRANT_LICENSE()` and `SYS$LOOKUP_LICENSE()`.

This is a temporary condition to help developers port their applications without LMF dependencies. In a future OpenVMS I64 release, we will enable LMF. At that time, product authorization keys (PAK) will be required.

NOTE If you have a product that calls LMF looking for a PAK that you do *not* expect to be loaded, a failure status will not be generated. Rather, due to the current workaround, LMF returns “success”. For this release, please temporarily comment out the applicable LMF code.

Error Log Report Formatter (ERF)

The Error Log Report Formatter (ERF) has not been ported to OpenVMS I64 Evaluation Release Version 8.1. There are no plans to port this facility to future versions of I64.

SORT32 and Hypersort Release Notes

This section contains release notes for SORT32 and Hypersort for OpenVMS I64 Evaluation Release Version 8.1. The following versions are available:

- SORT32 V08-007
- Hypersort V08-002

SORT32 and Hypersort -- Problems Corrected

SORT32 and Hypersort no longer fail with ACCVIOs, OPCDECs, or hang in the following cases:

- NL: as the output device
- MERGE with fixed length records
- Variable length records
- SORT and MERGE from COBOL programs
- Hypersort with large (>4GB) files

SORT32 Work Files - Known Problem

SORT32 does not always delete temporary work files.

SORT/SPECIFICATION and Compound Conditions - Restriction

SORT32 does not issue a diagnostic message for a compound condition in a key specification file not enclosed in parentheses, such as the following:

```
/Condition=(Name=Test1, TEST=(Field2 EQ "X") AND (Field3 EQ "A"))
```

That condition should instead be specified as:

```
/Condition=(Name=Test1, TEST=((Field2 EQ "X") AND (Field3 EQ "A")))
```

SORT32 and CONVERT with DFS-served disks - Restriction

SORT, MERGE, and CONVERT with the output directed to a UNIX-served DFS-mounted disk return %SORT-E-BAD_LRL.

To workaround this restriction, write the output file to an OpenVMS disk and then copy the output file to the UNIX-served DFS-mounted disk, or use CONVERT/FDL where the FDL specifies the required LRL for the output file.

SORT32 Work File Directories - Restriction

SORT32 work files must be redirected to directories that allow multiple file versions that can handle the number of requested work files.

SORT32 and Hypersort Performance - Working Set Extent and Page File Quota

SORT32 and Hypersort use different sorting and work file algorithms. Either sort utility may be faster depending on the input file and the memory/disk/CPU configuration. Make sure that working set extent is at most one third of page file quota with either SORT32 or Hypersort.

SORT32 Performance - Variable Length Records

SORT32 allocates fixed-sized slots for sort work files based on the longest record length (LRL) information in the input file(s). To improve performance, try to set LRL information in the input file(s) as close as possible to the actual longest record length. Poor initial performance may be the result of sorting some files produced by C programs, because the LRL is set higher than needed (to 32767).

SORT32 and Hypersort - Reporting problems

If you find a problem with SORT or MERGE, execute the following commands:

```
$ WRITE SYS$OUTPUT "WSEXTENT =''F$GETJPI("","WSEXTENT")'"

$ WRITE SYS$OUTPUT "PGFLQUOTA=''F$GETJPI("","PGFLQUOTA")'"
$ SHOW LOGICAL SORTSHR
$ SORT/STAT (or MERGE/STAT)
```

Include the output in your problem report along with the input file(s) that demonstrate the problem.

DECram for OpenVMS

As of OpenVMS I64 Evaluation Release Version 8.1, the DECram for OpenVMS product is a system integrated product (SIP) that ships as part of the OpenVMS base operating system.

For the OpenVMS I64 systems, DECram is available for use without the need of a DECram license. The next release of OpenVMS I64 will require a license to use the DECram Product.

Machine Check Aborts and Processor Inits on the rx4640

Machine check abort handling and processor init handling are not supported on the rx4640. If a machine check abort or processor init occurs on the primary processor in an rx4640, the system hangs. If a machine check abort or processor init occurs on a secondary processor in an rx4640, the system either hangs or crashes with either a CPUSANITY or CPUSPINWAIT bugcheck code. Because using the TC command from the Management Processor (MP) console generates processor inits, do not use this command on the rx4640.

System Management Release Notes

The following sections describe system management release notes.

New System Parameter - VHPT_SIZE

The VHPT_SIZE system parameter is a new parameter for I64 systems only. VHPT_SIZE is the number of kilobytes to allocate for the Virtual Hash Page Table (VHPT) on each CPU in the system:

- Zero (0) indicates that no VHPT is allocated.
- One (1) indicates that OpenVMS is to choose a default size that is appropriate for your system configuration.

If a VHPT is created, the smallest size is 32 KB. The VHPT_SIZE must be a power of 2 KB in size. If the number specified is not a power of 2, OpenVMS chooses a VHPT size to use for your system that is close to the number specified.

If insufficient memory is available during system startup, OpenVMS might choose a smaller size for the VHPT of each CPU.

Table 2-1 provides a summary of possible values for VHPT_SIZE.

Table 2-1 Summary of Possible Values for VHPT_SIZE

Value	Description
0	Do not create a VHPT on each CPU.
1	OpenVMS chooses a VHPT of an appropriate size for each CPU. (default)
n	Create a VHPT of nKB for each CPU, where n is a power of 2 that is 32 or greater (for example, 64 KB.) (The maximum value, however, is platform-dependent)

SYSGEN, SYSMAN, SDA, and SYSBOOT Changes

SYSGEN, SYSMAN, SDA, and SYSBOOT no longer display obsolete system parameters unless you use the new /OBSOLETE qualifier on the SHOW command or you use an exactly matching name in a SHOW parameter-name command.

- SHOW/OBSOLETE – Displays all obsolete system parameters
- SHOW <parameter_name> – Displays OBSOLETE for any specified parameter that is obsolete.

System Parameter Changes

Obsolete System Parameters

The following system parameters are now marked as obsolete:

BJOBLIM
BOOT_STYLE
EXUSRSTK
LAMAPREGS
NISCS_LAN_OVRHD
NJOBLIM
QBUS_MULT_INTR
SA_APP
SBIERRENABLE
SCSCONNECT
SD_ALLOCLASS
TAILORED
UDABURSTRATE

VECTOR_MARGIN
VECTOR_PROC
XFMAXRATE

System Parameter Updates

In the OpenVMS I64 Evaluation Release Version 8.1, a number of system parameters have been updated to provide better default, minimum, and maximum values for I64 systems.

- Support for 32,767 processes

  OpenVMS now supports a maximum of 32767 processes. The maximum values of the following system parameters have been increased to support this change:

  BALSETCNT — MAX=32,765
  IJOBLIM — MAX=32,765
  MAXPROCESSCNT — MAX=32,767

- Non-paged pool lookaside list trimming now turned off

  Trimming of the non-paged pool lookaside lists is now turned off by default. Trimming of these lists can result in a fragmented non-paged pool variable list. In some cases, a heavily fragmented variable pool list might result in degraded system performance.

  NPAG_AGGRESSIVE DEF=100
  NPAG_GENTLE DEF=100

- Increase in minimum value of CTLPAGES

  The minimum value of CTLPAGES has been increased to a value that can support a minimum boot:

  CTLPAGES MIN=64

- Larger CLISYMTBL values

  Both the maximum and minimum values of CLISYMTBL have been increased to support the creation of additional and large DCL symbols. In addition, the minimum value of CLISYMTBL has been increased to support a minimum boot:

  CLISYMTBL MAX=8,192 MIN=64

- Larger default values of CHANNELCNT

  The default values of CHANNELCNT have been increased to reduce the failure of applications that have large number of files open because of an insufficient CHANNELCNT value:

  CHANNELCNT DEF=512 MIN=64

- Parameters associated with global sections increased

  System quotas associated with Global Section have been increased to provide more appropriate default values:

  GBLPAGFIL DEF=16,394
  GBLSECTIONS DEF=1,024

- Default value of DUMPSTYLE changed for a compressed/selective dump

  The default value of the DUMPSTYLE parameter has been changed to produce a compressed/selective dump. This change helps ensure that necessary data is recorded in the system dump file when a system has a small system dump file.

  DUMPSTYLE DEF=9

- Default and minimum values of QUANTUM decreased

With faster systems, the amount of work a CPU can do in a given amount of time has increased. Lowering the default value of QUANTUM allows a CPU to schedule more processes per second.

QUANTUM DEF=5 MIN=1

- Values of the GH_* parameters increased

  The maximum value of the GH_* parameters has been increased to 65,536 (512MB). Note that it is impossible to boot a system with all of these parameters increased to the MAX value. These parameters consume a 32-bit address space, which is limited to 2GB.

  I64 images have a read-only segment that can be shared. To support this, the GH_RES_DATA parameter default has been increased from 0 to 100 pages.

  GH_EXEC_CODE MAX=65,536
  GH_EXEC_DATA MAX=65,536
  GH_RES_CODE MAX=65,536
  GH_RES_DATA MAX=65,536 DEF=100
  GH_RSRVPGCNT MAX=65,536

- Default value of WSMAX increas

  The default value of WSMAX has been increased to support the large physical memory on I64 systems:

  WSMAX DEF=131,072

- PQL parameters increased

  The PQL parameters are used to provide default and minimum values for various quotas when detached processes are created. A number of these parameters have been increased to provide more reasonable defaults and minimums to accommodate the large physical memory on I64 systems:

  PQL_DBYTLM DEF=262,144
  PQL_MBYTLM DEF=65,536
  PQL_DPGFLQUOTA DEF=262,136
  PQL_MPGFLQUOTA DEF=8,192
  PQL_DWSDEFAULT DEF=8,192
  PQL_MWSDEFAULT DEF=4,096
  PQL_DWSQUOTA DEF=16,384
  PQL_MWSQUOTA DEF=8,192
  PQL_DWSEXTENT DEF=65,536
  PQL_MWSEXTENT DEF=16,384
  PQL_DENQLM DEF=2,048
  PQL_MENQLM DEF=64

- Defaults for paged and non-paged pool increased

  The default values for a number of the paged and non-paged pool parameters have been increased for Itanium:

  NPAGEDYN DEF=4,194,304 (4MB)
  NPAGEVIR DEF=16,777,216 (16MB)
  PAGEDYN DEF=4,194,304 (4MB)

- Default of the system working set quota increased

  The default value for SYSMWCNT has been increased.

  SYSMWCNT DEF=8,192

3 Programming Release Notes

This chapter provides release notes about programming on OpenVMS I64 Version 8.1 systems.

Programs Need to Be Recompiled for Version 8.1

Because OpenVMS I64 Version 8.0 and 8.1 are considered evaluation releases, HP reserves the right to make incompatible changes to procedures, data structures, and programming interfaces. Because of this, all programs compiled for OpenVMS Version 8.0 must be recompiled after installing OpenVMS I64 Version 8.1.

The following areas have changed:

- Shareable image interfaces have changed so that they are not compatible.
- The image and object format has changed from ELF ABI Version 1 to ELF ABI Version 2.
- The debugging information in images has changed so that it is not compatible.
- The compilers available with this release have improved code generation sequences.

When the production quality release of the OpenVMS I64 operating system is available, our normal upward compatibility policy will go into effect.

Removed System Services

The SYS$GOTO_UNWIND system service is not available on OpenVMS I64 systems; recode to use SYS$GOTO_UNWIND_64. SYS$GOTO_UNWIND was removed to assure applications are ported correctly. See the *Guide to Porting Applications from OpenVMS Alpha to OpenVMS Industry Standard 64 for Integrity Servers* for more information. This new service, SYS$GOTO_UNWIND_64, exists on OpenVMS Alpha Version 7.3-2, as well as OpenVMS I64 Evaluation Release Version 8.1, so applications can use common code. See “New and Revised System Services” on page 47, for more information about SYS$GOTO_UNWIND_64.

New and Revised System Services

The following table summarizes new and changed system services for OpenVMS I64 Version 8.1. For more information about these changes, see the *HP OpenVMS System Services Reference Manual*.

Table 3-1 New and revised system services

System Service	Documentation Update
SYS$CREATE_GPFN	The new flag SEC$M_UNCACHED has been added.
SYS$CRMPSC_GPFN_64	The new flag SEC$M_UNCACHED has been added.
SYS$CRMPSC_PFN_64	The new flag SEC$M_UNCACHED has been added.
SYS$GOTO_UNWIND_64	New service
SYS$SET_RETURN_VALUE	New service
SYS$GETDVI	The following new item codes have been added:
	DVI$_SHDW_MBR_TWO
	DVI$_SHDW_COPIER_NODE
	DVI$_SHDW_DEVICE_COUNT
	DVI$_SHDW_GENERATION
	DVI$_SHDW_MASTER_MBR
	DVI$_SHDW_MBR_COPY_DONE
	DVI$_SHDW_MBR_COUNT
	DVI$_SHDW_MBR_MERGE_DONE
	DVI$_SHDW_MINIMERGE_ENABLED
	DVI$_SHDW_MBR_READ_COST
	DVI$_SHDW_READ_SOURCE
	DVI$_SHDW_TIMOUT
	DVI$_QLEN
SYS$MGBLSC_GPFN_64	The new flag SEC$M_UNCACHED has been added.
SYS$MOUNT	The MT3 definition MNT$_DENSITY has been added.
SYS$INIT_VOL	The MT3 definition INIT$DENSITY has been added.
SYS$GETRMI	The following new item codes have been added:
	RMI$_CPUEXEC
	RMI$_CPUIDLE
	RMI$_CPUINTSTK
	RMI$_CPUKERNEL
	RMI$_CPUMPSYNCH
	RMI$_CPUSUPER

Table 3-1 New and revised system services (Continued)

System Service	Documentation Update
	RMI$_CPUUSER
SYS$LKWSET	Revised to reflect new behaviors of Alpha and I64 systems.
SYS$LKWSET_64	Revised to reflect new behaviors of Alpha and I64 systems.
SYS$SETFLT	The new flag, FLT$M_EXECUTABLE, has been added.
SYS$SETFLT_64	The new flag, FLT$M_EXECUTABLE, has been added.
SYS$SETSTK_64	New service
SYS$ULWSET	Revised to reflect new behaviors of Alpha and I64 systems.
SYS$ULWSET_64	Revised to reflect new behaviors of Alpha and I64 systems.

New System Services

This section contains information about new system services for OpenVMS I64 Version 8.1. See the *HP OpenVMS System Services Reference Manual* for more information.

SYS$GOTO_UNWIND_64 (Alpha and I64 Only)

Description

Unwinds the call stack.

Format

```
SYS$GOTO_UNWIND_64 target_invo ,target_pc ,NewRetVal , NewRetVal2
```

C prototype

```
int sys$goto_unwind_64(void *target_invo_64, void *(*(target_pc_64), unsigned_int64
*newretval, unsigned_int64 *newretval2);
```

SYS$SET_RETURN_VALUE (Alpha and I64 Only)

Description

The Set Return Value service allows the caller to specify the values in the Mechanism Array. It can also be used to set a condition code.

Format

```
SYS$SET_RETURN_VALUE mechanism_arg,return_type,return_value
```

C prototype

```
int sys$set_return_value(void *mechanism_arg, unsigned int *return_type, void *return_value);
```

SYS$LKWSET and SYS$LKWSET_64 System Services

If your application uses SYS$LKWSET or SYS$LKWSET_64 to lock itself into memory, consider replacing these calls with calls to the new LIB$LOCK_IMAGE RTL routine. Similarly, replace the SYS$ULWSET and SYS$ULWSET_64 calls with calls to the new LIB$UNLOCK_IMAGE RTL routine.

For more information about these services, see the *HP Guide to Porting Applications from OpenVMS Alpha to OpenVMS Industry Standard 64 Integrity Servers*.

PFN-Mapped Sections

Mapped I/O space on an OpenVMS I64 system may require non-cached access. You must set the SEC$M_UNCACHED flag when a PFN-mapped section is created if this section must be treated as uncached memory. The following system services now accept this flag:

- SYS$CRMPSC_PFN_64
- SYS$CREATE_GPFN
- SYS$CRMPSC_GPFN_64

In addition, the SYS$MGBLSC_GPFN_64 service accepts, but ignores the flag. The cached/uncached characteristic is stored as a section attribute, and the system uses this attribute when the section is mapped.

On OpenVMS Alpha systems, all four services accept but ignore the SEC$M_UNCACHED flag.

Note that the older services, SYS$CRMPSC and SYS$MGBLSC were not updated and do not accept the new flag.

New RTL LIB$ Routines

The following information describes the two new Run-Time Library (RTL) LIB$ routines, LIB$LOCK_IMAGE and LIB$UNLOCK_IMAGE.

LIB$LOCK_IMAGE

The LIB$LOCK_IMAGE routine locks the specified image in the process's working set.

Format LIB$LOCK_IMAGE *address*

Returns VMS usage: cond_value
type: longword (unsigned)
access: write only
mechanism: by value

Arguments *address*
VMS usage: address
type: quadword (unsigned)
access: read only
mechanism: by value

Address of a byte within the image to be locked in the working set. If the address argument is 0, the current image (which contains the call to LIB$LOCK_IMAGE) is locked in the working set.

Description

LIB$LOCK_IMAGE locks the specified image in the process's working set.

This routine is typically used by a privileged user before the program, executing in kernel mode, raises IPL above IPL 2. Above IPL 2, paging is not allowed by the system. The program must access only pages valid in the process's working set.

Condition Values Returned

SS$_WASSET The specified image is locked in the working set and had previously been locked in the working set.

SS$_WASCLR The specified image is locked in the working set and had previously not been locked in the working set.

Other status codes returned by sys$lkwset_64.

LIB$UNLOCK_IMAGE

The LIB$UNLOCK_IMAGE routine unlocks the specified image from the process working set.

Format LIB$UNLOCK_IMAGE *address*

Returns
VMS usage: cond_value
type: longword (unsigned)
access: write only
mechanism: by value

Arguments *address*
VMS usage: address
type: quadword (unsigned)
access read only
mechanism: by value

Address of a byte within the image to be unlocked from the working set. If the address argument is 0, the current image (that contains the call to LIB$LOCK_IMAGE) is unlocked from the working set.

Description

LIB$UNLOCK_IMAGE unlocks the specified image from the process's working set.

This routine is typically used by a privileged user after the program, executing in kernel mode, lowers IPL to 0 or 2. Above IPL 2, paging is not allowed by the system. The program must access only pages valid in the process's working set. LIB$LOCK_IMAGE is used to lock the image in the working set.

Condition Values Returned

SS$_WASSET The specified image is unlocked from the working set and had previously been locked in the working set

SS$_WASCLR The specified image is unlocked from the working set and had previously not been locked in the working set

Other status codes returned by sys$ulwset_64.

Linker Release Notes

New Linker Qualifier: /BASE_ADDRESS

The base address is the starting address that you want the linker to assign to an executable image. The linker assigns this address to the first program segment it creates. This qualifier is different from the BASE= option for VAX systems (illegal on Alpha and I64 systems), as it can not be used to base shareable images. It replaces the CLUSTER=,[base-address] option on Alpha, which is illegal on I64.

The /BASE_ADDRESS qualifier has the following format:

`/BASE_ADDRESS=`*`address`*

where *address* is the location at which you want the executable image based. You can express this location as decimal (%D), octal (%O), or hexadecimal (%X) numbers. The default is hexadecimal.

This qualifier works for all executable images and is accepted but ignored for shareable images, where the image activator determines the address of the shareable image at run time.

New Linker Option: SHORT_DATA

The SHORT_DATA option allows you to combine read-only and read-write short data sections into a single segment.

Normally, there are two short data segments that the image activator relocates and puts together that are accessed via offsets to the GP register. The `SHORT_DATA=WRT` option sets all short data program section attributes to WRT, including the linker-generated, read-only $LINKER SDATA$. This allows the linker to collect all short data program sections into a single, writable short data segment. In this case, $LINKER SDATA$ no longer starts at a page boundary and the linker reclaims unused space between the former read-only and read-write segments. When the short data sections are merged into a single segment, $LINKER SDATA$ is aligned on a 16 byte boundary.

Because short data segments are limited to 4 MBs, setting `SHORT_DATA` to WRT can eliminate the problem of overflow. Using this option reclaims the unused space between the two segments that can total up to 65,535 bytes.

The SHORT_DATA option has the following format:

`SHORT_DATA=[NO]WRT`

Where `SHORT_DATA` sets all short data program section attributes to WRT or NOWRT. This affects the collection of program sections, which are now placed in one writable or readonly short data segment.

CAUTION When setting SHORT_DATA to WRT, do not write to formerly read-only data because this data is no longer protected.

Initialized Overlaid Program Sections Handled Differently on I64 Systems

The following condition produces different results on OpenVMS I64 systems than on Alpha systems:

Two program sections, each declaring two longwords, are overlaid. The first program section initializes the first longword and the second program section initializes the second long word.

On Alpha systems, the linker produces an image section with the first and second longwords initialized. The linker uses the same start address for the program sections so they occupy the same virtual memory and allocate enough space to accommodate the largest of the program sections.

On I64 systems, the linker produces an image segment where either the first or second longword is initialized, depending on the order in which they were linked. Because the linker does not read all overlaid initialized program sections, and because these initializations can be made with identical values, the linker does not produce an error message for this condition. This will be corrected in a future release of the linker. The linker will read the sections and will issue a warning when the initializations are different.

Process Quota May Be Exceeded When Linking a Large Number of Objects in an Options File

During a link operation, the linker may exceed your process quotas. For example:

```
RMS-E-ACC, ACP file access failed
-SYSTEM-F-EXQUOTA, process quota exceeded
```

If this condition happens, ask your system administrator to increase one or more of the following quotas: Open File Limit, Buffered I/O Byte Count Limit and Page File Limit. The following command can be used:

```
$ SET PROCESS/PRIVILEGE=SYSPRV
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> MODIFY your_username/FILLM=500/BYTLM=250000/PGFLQUOTA=250000
```

In the above example, the values for the /FILLM, /BYTLM, and /PGFLQUOTA switches are based on the user requesting a quota increase. When linking an extremely large image, you might need to increase these values. In most cases, simply increasing the Open File Limit might be enough.

Note that the I64 linker maps files to P2 space but does not close the files. As a result, no matter how many objects and images you are linking against, the system provides plenty of P2 space. However, the Open File Limit set for your process may be too low to account for the number of files that the linker has open. The Buffered I/O Byte Count Limit and the Page File Limit may be related to operations that the linker has with the new object file format, the Executable and Linkable Format (ELF).

For links of very large images, you may get the following message:

-SYSTEM-F-SECTBLFUL, processor global section table is full

When this limit is reached, the linker has used too many process sections when mapping files to P2 space. As a workaround, reset the number of process sections (PROCSECTCNT) and reboot. For example, if you are linking 1,950 files, reset the number of process sections to 2,000, as follows:

```
SYSBOOT> DISABLE CHECKS
SYSBOOT> SET PROCSECTCNT 2000
SYSBOOT> CONTINUE
```

NOTE These limitations are temporary. In a future release of the linker, the linker will close and reopen files.

Process Quota May Be Exceeded When Mapping a Large Number of Object Modules

When using the linker, be aware that the librarian that is invoked contains temporary code that copies modules instead of mapping them directly. As a result, if a large number of library modules are mapped, your process quotas may be exceeded. Contact you system manager to increase your quotas.

The quota likely exceeded is the Paging File Quota (PGFLQUOTA). Other quotas that may be exceeded are the File Limit quota (FILLM) and the Byte Limit quota (BYTLM) (See the release note titled "Process Quota May Be Exceeded When Linking a Large Number of Objects in an Options File" for instructions.)

The LBRSHR image for OpenVMS I64 systems does not directly map modules from the library file when the linker calls the LBR$MAP_MODULE routine. Instead, LBRSHR creates a region in P2 space, transfers the module's contents from the library file to the newly created region, and returns the address of the newly created area to the linker.

New Alignments for PSECT_ATTRIBUTE Option

The PSECT_ATTRIBUTE option now accepts integers 5, 6, 7, and 8 for the alignment attribute. The integers represent the byte alignment indicated as a power of 2. (For example, 2 ** 6 represents a 64-byte alignment.) The keyword HEXA (for hexadecimal word) was added for 2 ** 5 (that is, a 32-byte or 16-word alignment).

Specifying Alignment with the PSECT_ATTRIBUTE Option

Do not use the PSECT_ATTRIBUTE to specify a smaller section alignment than the alignment that the compiler gave to the section. If you specify a smaller alignment for a program section than any compiler-assigned alignment from all contributions to this section, the linker now issues a warning. For example:

```
$ link hi,sys$input/opt

Linker T01-35

Librarian T01-08

psect_attr=$literal$,byte

%ILINK-W-CONFALGN, option alignment attribute (1) is less than compiler contribution (16)

        for section $LITERAL$, module HI, file EVMS$:[IA64_LINKER.SRC]HI.OBJ;2, option ignored.
```

For the VAX and Alpha systems, the linker inappropriately aligns the program section on the boundary that you specified ("byte", in the above code example), and places all the contributions to that program section (from other modules you might have linked with "Hi", in the above example) on boundaries that were not specified by the compiler. The linker would not have issued an error message.

The linker should always align sections on the boundary specified by the compiler.

The PSECT_ATTRIBUTE option aligns the section on the specified boundary when it is equal to or greater than that which the compiler specified. It does not align each individual contribution to the section; rather, the total section. The PSECT_ATTRIBUTE option follows the compiler's alignment specification when it aligns each individual contribution.

Conventions for Specifying Image Names

The following conventions describe the various names that apply to an image:

•Images are given an image file specification (for example, Foo.exe) that can be changed with the DCL RENAME command.

•The image name is specified with the NAME= option and stored in the image in a note of type NT_VMS_IMGNAM. This name can be different than the image file specification name. However, if you do not use the NAME= option, the name defaults to the image file specification name. The Analyze utility displays this name as the "Image name". You cannot change this name with the DCL RENAME command.

•An additional name for the image is associated with the Global Symbol Table (GST) and stored in the image in a note of type NT_VMS_GSTNAM. The linker sets this name to be the same as the image file specification name. This name is used by the librarian when you insert an image in an image library. It is displayed by the Analyze utility as the "Global Symbol Table Name". You cannot change this name with the DCL RENAME command.

On Alpha systems, the image name specified with NAME = is used to identify self-references in the shareable image list (that is, calls from within the image to itself, by way of an alias name in the symbol vector).

On I64 systems, there is no entry in the shareable image list for the current image. Self-references are referred to with a special index value (-1 in the DT_VMS_FIXUP_NEEDED) into the shareable image list (that is, a set of DT_NEEDED entries).

Linkage Checks on Shareable Image Alias Symbols Are Not Performed

For procedures with symbol vector aliases, the linker currently does not propagate register linkage information from the symbol to its symbol vector alias. This prevents linkage checks from being performed between the caller of the routine and the target routine.

For example, in the following option file line for a shareable image, the symbol BASE_SYMBOL retains its register linkage information when it is placed in the shareable image's Global Symbol Table (GST). However, the symbol ALIAS is marked to have its register linkage checks ignored.

```
SYMBOL_VECTOR = ( ALIAS / BASE_SYMBOL = PROCEDURE )
```

As a result, if there are register conflicts between the caller of ALIAS and the routine ALIAS (for example, BASE_SYMBOL), the linker does not find these conflicts.

Changes to Linker Map File for OpenVMS I64 Systems

The information in the linker map file has changed and expanded for the current release. See Appendix B, "Linker Maps," on page 121, for the linker maps which illustrates these changes.

Flags Now Set When /TRACEBACK, /DEBUG and /DSF Qualifiers Are Specified

When the /TRACEBACK, /DEBUG and /DSF linker qualifiers are specified, the following flags are set in the dynamic segment under the tag value DT_VMS_LNKFLAGS. The meanings of /TRACEBACK, /DEBUG and /DSF have not changed from Alpha, but the meanings and the names of the flags have changed.

Flag	**Meaning**
VMS_LF_IMGSTA	Image execution is to begin by calling SYS$IMGSTA. The image activator includes SYS$IMGSTA as the first address in the (traditional OpenVMS style) transfer vector.
VMS_LF_CALL_DEBUG	SYS$IMGSTA checks this flag to determine whether it calls the debugger.
VMS_LF_TBK_IN_IMG	Traceback records are present in the image file
VMS_LF_DBG_IN_IMG	Debug information is present in the image file
VMS_LF_TBK_IN_DSF	Traceback records are present in the DSF file
VMS_LF_DBG_IN_DSF	Debug information is present in the DSF file

The flags will be set according to the following table:

Qualifiers	**IMGSTA**	**CALL_ DEBUG**	**TBK_IN_ IMG**	**DBG_IN_ IMG**	**TBK_IN_ DSF**	**DBG_IN_ DSF**
/NoTrace /NoDebug /NoDSF	0	0	0	0	0	0
/Trace /NoDebug /NoDSF	**1**	0	**1**	0	0	0
/NoTrace **/Debug** /NoDSF	**1**	**1**	**1**	**1**	0	0
/Trace /Debug /NoDSF	**1**	**1**	**1**	**1**	0	0
/NoTrace /NoDebug **/DSF**	0	0	0	0	**1**	**1**
/Trace /NoDebug **/DSF**	**1**	0	**1**	0	**1**	**1**
/NoTrace **/Debug /DSF**	**1**	**1**	**1**	0	**1**	**1**
/Trace /Debug /DSF	**1**	**1**	**1**	0	**1**	**1**

Notes:

The value of SYS$IMGSTA is no longer included in the image's transfer array; only a flag that indicates it is to be called. The image activator knows where the value of SYS$IMGSTA is located.

These flags do not appear in a DSF file. DSF files have no dynamic segment and therefore no DT_VMS_LNKFLAGS field.

The Linker no longer supports /DEBUG=*filename* on I64 systems. Link alternative debuggers as a separate image and then define LIB$DEBUG to point to that image. SYS$IMGSTA always calls whatever is pointed to by LIB$DEBUG.

When /DSF is specified, along with /TRACEBACK or /DEBUG, the VMS_LF_TBK_IN_IMG (traceback in image) flag will be set. This is a change in behavior from Alpha, where traceback records were not included in the image when /TRACEBACK/DSF or /DEBUG/DSF was specified. Note that debugger records do not get copied to an image whenever /DEBUG/DSF is specified. /DEBUG causes the IMGSTA bit to be set in the image, but does not cause the debugger records to be copied into the image.

The following table indicates where global symbol definitions are written during a link operation using the debugging qualifiers.

Qualifiers	Global Symbols in Image	Global Symbols in DSF File
/NoTrace /NoDebug /NoDSF	0	0
/Trace /NoDebug /NoDSF	**1**	0
/NoTrace **/Debug** /NoDSF	0	**1**
/Trace /Debug /NoDSF	**1**	0
/NoTrace /NoDebug **/DSF**	0	**1**
/Trace /NoDebug **/DSF**	0	**1**
/NoTrace **/Debug /DSF**	0	**1**
/Trace /Debug / DSF	0	**1**

Using Mixed-Case Character Module Names During a Link Operation

Object module libraries can have modules with names containing mixed-case characters.

The linker processes an options file in a case-insensitive manner and changes all characters to uppercase before processing begins. To get the linker to process a module with a mixed-case module name from a library, you must first set the linker to case-sensitive mode. This is important for I64 modules because the module names are mixed case by default.

Use the CASE_SENSITIVE=YES option to process a module with mixed-case names. This option causes the linker to perform case-sensitive reads of all characters to the right of the equal sign (=) and uppercase all characters up to the left of the first equal sign (=). To turn case sensitivity off, specify NO (in all capital letters) for the CASE_SENSITIVE= option.

Example:

```
case=Yes

My_Lib/library/include=(Add_Func,Sub_Func)
symbol_vector=(Add_Func=PROCEDURE,PAGE_COUNT=DATA)
case=NO
```

The linker processes:

```
CASE=YES
MY_LIB/LIBRARY/INCLUDE=(Add_Func,Sub_Func)
SYMBOL_VECTOR=(Add_Func=PROCEDURE,PAGE_COUNT-DATA)
CASE=NO
```

Understanding Linkage Messages

Some HP language processors produce call linkage information that verifies the linkage of a caller or a called routine to determine consistent use of the general registers. Linkage information is supplied for only the general registers 0 through 31. Conflicts arise when the linkage information indicates a problem.

The following example shows a warning message that displays when the linker detects a linkage conflict:

```
%ILINK-W-LNKGERR, linkage to routine X is not compatible with linkage of caller
-ILINK-I-LNKGFRM,     calling module MODSRC file DISK:[DIR]SOURCE_CALL.OBJ;1
-ILINK-I-LNKGTO,      target  module MODTRG file DISK:[DIR]TARGET.OBJ;1
%ILINK-I-LNKGTYP,     source type of JSB to target type of CALL
%ILINK-I-LNKGREG,     register AI not provided (needed at target)
%ILINK-I-LNKGREG,     Itanium register R19 (call=OUTPUT; target=SCRATCH)
```

Informational messages follow the warning message to indicate the source and target of the call. Following these messages, one or more conflict indications of the condition that caused the warning message is displayed. For example, if a caller does not provide argument information in the argument information register (AI), when the target routine requires the AI, the linker will generate an informational message similar to the one in the above example.

In addition to missing required information, inconsistent or incompatible use of the general registers can also cause a linkage conflict. The linkage information contains register policies for the general registers, which are as follows:

Volatile : A register with this policy may not be used to pass information between procedures, either as input or output.

Scratch : A register with this policy may be modified by a procedure call.

Output : A register with this policy may be used to pass information back to the calling procedure

Preserve : A register with this policy must have its contents saved and then restored if it is to be used by the target routine.

The register policies of a standard linkage call are as follows:

Itanium Register	**Policy**
R2	Volatile
R3	Scratch
R4 - R7	Preserve
R8 - R9	Output
R10 - R11	Scratch
R12 - R18	Volatile
R19 - R24	Scratch
R26 - R31	Scratch

The following table indicates compatible policies between the caller's register policies (row heading) and the target routine's register policies (column heading):

Caller	**Target**			
	Volatile	Scratch	Output	Preserve
Volatile	X			
Scratch		X		X
Output			X	X
Preserve				X

In the sample message:

```
%ILINK-I-LNKGREG,          Itanium register R19 (call=OUTPUT; target=SCRATCH)
```

Note that in the sample message, Itanium register R19 has a caller register policy of "Output" but a target routine register policy of "Scratch". Based on the above table, R19 is not compatible between the caller and the target routines.

Additionally, the calling mechanism between the routines is not compatible. This occurs if the caller is performing a JumptoSuBroutine (JSB) to the target routine when the target routine is expecting a CALL mechanism to be used. The standard procedure call uses the CALL mechanism and provides the Global Pointer (GP) value and the AI register fill with the argument information.

Certain Itanium® registers must be defined to have specific policies. If these registers do not contain the correct policies, the linker will issue a warning indicating that either the target linkage is invalid (1) or a message of the caller is invalid (2):

```
(1)%ILINK-W-INVLNKG, invalid target linkage for symbol INV_LNKG
(2)%ILINK-W-INVRLNKG, invalid call linkage for symbol INV_LNKG
```

The following table lists the Itanium® general registers that must be of specific policies:

Itanium Register	**Policy**
R2	Volatile
R12-R18	Volatile

For more information, see the *OpenVMS Calling Standard* manual included on the documentation CD-ROM in the Version 8.1 kit.

Explanation of Potential Conflicts that Result in Linkage Messages

Some conflicts arise when performing cross language calls (for example, MACRO-32 to BLISS), where the calling standards for OpenVMS differ between Alpha and Itanium. There are fewer registers preserved by default with the Itanium® version of the calling standard. Because of this, there may be assumptions about registers that are saved at the target routine and registers that are not saved.

Linkage statement mismatches also cause the linker to display warning messaages. For example, if a calling routine is expecting registers R12-R15 (Alpha register numbers) to be preserved but the target does not preserve them, the linker will list registers R20, R21, R30, and R31 as having a linkage problem. Note that the linker will display Itanium register numbers which, for the above statement, are registers R14, R15, R12, and R13 for Alpha respectively.

Additionally, a problem can exist when registers that are not properly declared for the function they are serving (for example, declaring registers to return values as NOPRESERVE rather than OUTPUT.)

To correct these potential conflicts, examine the declaration and definition linkages for all cases the linker indicates and correct these problems to ensure a clean link operation. Because the linker considers these conflicts warnings, the completion code status of the image will not be set to SUCCESS. For a shareable image, this means that images linked against an image with linkage issues will receive a completion status other than SUCCESS.

Linker Appears to Hang When Many Files are Specified

When the RMS_RELATED_CONTEXT linker option is on (the default is RMS_RELATED_CONTEXT=YES) and a nonexistent file is specified in a list of files for the LINK command, the linker's call to LIB$FIND_FILE takes a long time to complete, and the linker may appear to hang. Depending on the number of files being linked and the use of logical names in their specification, the linker may take hours to finish because LIB$FIND_FILE locates every combination of the missing file's prefix before displaying a "file not found" message.

NOTE You cannot terminate the linker processes with Ctrl/Y while the linker is calling LIB$FIND_FILE.

Workaround

In the options file, set the call to RMS_RELATED_CONTEXT as follows:

RMS_RELATED_CONTEXT=NO

With the RMS_RELATED_CONTEXT option set to NO, a missing file in the linker command generates an immediate "file not found' message.

Because RMS_RELATED_CONTEXT=NO no longer provides name stickiness, you must specify the fully-qualified file name. For example, consider the following LINK command:

```
$ LINK DSK:[TEST]A.OBJ, B.OBJ
```

When RMS_RELATED_CONTEXT is set to NO, you must specify the following command:

```
$ LINK DSK:[TEST]A.OBJ, DSK:[TEST]B.OBJ
```

For more information about the RMS_RELATED_CONTEXT option, refer to the *OpenVMS Linker Utility Manual*.

Example

The following example shows how the linker appears to hang when the file DOES_NOT_EXIST.OBJ is included in the list and the RMS_RELATED_CONTEXT option is not specified (and, therefore, defaults to YES).

```
$ DEFINE DSKD$  WORK4:[TEST.LINKER.OBJ.]
$ DEFINE RESD$   ROOT$,  ROOT2$,  ROOT3$,  ROOT4$,  ROOT5$,  DISK_READ$:[SYS.]    [1]
$ DEFINE ROOT$,  WORK4:[TEST.PUBLIC.TEST]
$ DEFINE ROOT2$  WORK4:[TEST.LINKER.]
$ DEFINE ROOT3$  WORK4:[TEST.UTIL32.]
$ DEFINE ROOT4$  WORK4:[TEST.PUBLIC.]
$ DEFINE ROOT5$  WORK4:[TEST.PUBLIC.TMP]
$ LINK/MAP/FULL/CROSS/EXE=ALPHA.EXE   RESD$:[TMPOBJ]  A.OBJ -
_$ RESD$:[SRC]B.OBJ,C,DSKD$:[OBJ]D.OBJ,E,RESD$:[TMPSRC]F.OBJ, -
_$ RESD$:[TEST]G.OBJ,RESD$:[SRC.OBJ]H,RESD$:[COM]DOES_NOT_EXIST.OBJ
<Crtl/T>
```

```
NODE::_FTA183:  15:49:46  LINK CPU=00:02:30.04  PF-5154  IO-254510  MEM=134   [2]
<Crtl/T>
NODE::_FTA183:  15:49:46  LINK CPU=00:02:30.05  PF=5154 IO=254513 MEM=134
<Crtl/T>
NODE::_FTA183:  15:50:02  LINK CPU=00:02:38:27  PF=5154 IO=268246 MEM=134
<Crtl/T>
NODE::_FTA183:  15:50:02  LINK CPU=00:02:38.28  PF=5154 IO=268253 MEM=134
<Crtl/T>
NODE::_FTA183:  15:50:14  LINK CPU=00:02:44.70  PF=5154 IO=278883 MEM=134
<Crtl/T>
```

Callouts:

1. This command defines logical names and equivalents
2. Each time you press Crtl/T, the CPU and IO values increase, but the MEM and PF values do not, indicating that LIB$FIND_FILE is being called.

As shown in the following example, setting RMS_RELATED_CONTEXT to NO causes the link operation to finish immediately. Because there is no name stickiness, you must use complete file specifications. Also, use an options file, because RMS_RELATED_CONTEXT applies to all the files that follow it.

```
$ DEFINE DSKD$  WORK:[TEST.LINK.OBJ.]
$ DEFINE RESD$   ROOT$,  ROOT2$,  ROOT3$,  ROOT4$,  ROOT5$,  DISK_READ$:[SYS.]
$ DEFINE ROOT$  WORK4:[TEST.PUBLIC.TEST.]
$ DEFINE ROOT2$  WORK4:[TEST.LINKER.]
$ DEFINE ROOT3$  WORK4:[TEST.UTIL32.]
$ DEFINE ROOT4$  WORK4:[TEST.PUBLIC.]
$ DEFINE ROOT5$  WORK4:[TEST.PUBLIC.TMP.]
$ LINK/MAP/FULL/  CROSS  /EXE=ALPHA.EXE SYS$INPUT:/OPTION
     RMS_RELATED_CONTEXT=NO
RESD$:[TMPOBJ]A.OBJ,RESD$:[SRC]B.OBJ,RESD$:[SRC]C,DSKD$:[OBJ]D.OBJ
   SKD$:[OBJ]E,RESD$:[TMPSRC]F.OBJ,RESD$:[TEST]G.OBJ
   RESD$:[SRC.OBJ]H,RESD$:[COM]DOES_NOT_EXIST.OBJ <Crtl/Z>
%ILINK-F-OPENIN, error opening DISK_READ$:[SYS.] [COM]DOES_NOT_EXIST.OBJ; as
input
-RMS-E-FNF, file not found
```

Temporary Workaround for %ILINK-E-NOT21BITS (PCREL21B) Errors

On OpenVMS I64 systems, some code PSECTs in excess of 16MB need special handling that is not yet provided by the current linker. This problem appears as a NOT21BITS error, for example:

```
$ LINK/MAP PCRELA, PCREL21BIG0,  PCREL21BIG1.  PCREL21BIG2,  PCREL21BIG3, -
            PCREL21BIG4,    PCREL21BIG5,  PCREL21BIG6,  PCREL21BIG7,  PCREL21BIG8, -
            PCREL21BIG9,    PCREL21BIGA, PCREL21BIGB, PCREL21BIGC, PCREL21BIGD, -
            PCREL21BIGE,   PCREL21BIGF,  PCRELZ
%ILINK-E-NOT21BITS,  PCREK21B relocation exceeded 21 bits
                  relocation for section .rela$CODE$, module PCRELA, file
USER:[JOE[PCRELA.OBJ;1
```

Workaround

Break up the large PSECT with the CLUSTER=option, which groups input object files and creates separate PSECTs.

Use the Program Section Synopsis in the linker map to determine which module contributions need to be removed and placed in another PSECT, so that the remaining code fits into 16 MB sections.

See Appendix B, “Linker Maps,” on page 121, for sample Linker maps.

C Run-Time Library Routines

The C run-time library routine kill() is not working in this release.

Enhancements to the Analyze Utility

The Analyze utility has been modified to display Executable and Linkable Format (ELF) object and image files. Using record formats and landmark values in the file, ANALYZE can determine the architecture type and whether the file is an object or image file. Although ANALYZE accepts the /OBJECT (default) and /IMAGE qualifiers, using either does not restrict the analysis to the specified file type. By using the default /OBJECT to analyze and image, ANALYZE correctly identifies itself in the output file as "Analyze Object File".

The output from ANALYZE, unless the /SELECT qualifier is specified, consists of a module synopsis followed by a detailed analysis of the specified or default portions of the module.

To instruct ANALYZE to analyze and display specific sections or section types, three qualifiers have been added:

- /SECTIONS – Analyzes and displays ELF section information.
- /SEGMENTS – Analyzes and displays ELF segment information (has no effect on object modules).
- /FLAGVALUES – Controls the display of individual flags within a flag field.

In addition, ANALYZE now creates DCL symbols for all selectable information with the /SELECT qualifier. The symbol names consist of the prefix ANALYZE$ and a descriptive name of the information they hold.

Summary Information Displayed by ANALYZE for I64 Modules

For object modules, the following information is displayed:

- Module architecture and type
- Module name
- Module version
- Module creation date and time
- Language processor creator
- Section synopsis
- ELF module header

For image files, the following information is displayed:

- Image architecture and type
- Image name
- Image indentification
- Creating linker identification

- Link date and time
- Transfer vector array
- Transfer vectors
- Dynamic segment table
- List of needed shareable images
- System version array
- Program segment synopsis
- Section synopsis
- ELF header

Order of Information Displayed in the Detailed Area

ANALYZE will display the module information and requested sections and segments in the following order:

- Module or image synopsis
- Segment headers (images only)
- Segment contents (images only)
- Section headers
- Section contents

New Qualifiers

New qualifiers have been added to ANALYZE: /SECTIONS, /SEGMENTS, /SELECT, and /FLAGVALUES. Each of these is described in the following sections.

/SECTIONS Qualifier Keywords

/SECTIONS [=(keyword [,…])]

•This qualifier is used to select individual sections or section types to display.

•If no values are specified, the default keyword is HEADERS.

•Note: This qualifier and its keywords can be used only to form an inclusion list of sections to be displayed. You cannot use a negative form of this qualifier (for example, /NOSECTIONS) to form an exclusion list.

/SECTIONS=HEADERS

The default keyword. ANALYZE will display the details of each section contained in the section header table.

/SECTIONS=ALL

This keyword will cause ANALYZE to display a detailed analysis of every section in the module. Note that this can generate a large amount of output.

/SECTIONS=NUMBERS=(number [,…])

•This keyword will allow individual sections to be displayed. The selected sections will have a detailed display of their header and their contents. An informational message is displayed for section numbers that do not exist in the module.

•One or more numeric values may be specified.

•Section numbers may be specified in decimal, octal (using the %O prefix), or hexadecimal (using the %X prefix).

/SECTIONS=NULL

All sections of type SHT_NULL will be displayed. There is no data associated with sections of this type.

/SECTIONS=PROGBITS

•All sections of type SHT_PROGBITS will be displayed, with one exception listed below.

•Formatting for these sections depends on the EXECINSTR flag (SHDR$M_SHF_EXECINSTR) in its section header. If this bit is set, the section data will be displayed as machine instructions. Otherwise it will be displayed as hexadecimal data.

•Exception: Unwind data is represented by two sections; an unwind table section of type SHT_IA_64_UNWIND and an unwind information section of type SHT_PROGBITS.

Unwind sections will be displayed if /SECTIONS=UNWIND is specified. Even though they are of type PROGBITS, they will not be displayed if /SECTIONS=PROGBITS is specified.

/SECTIONS=CODE

This keyword will cause display of all sections of type SHT_PROGBITS where the executable flag is set (SHDR$M_SHF_EXECINSTR in the section header). The section data will be displayed as machine instructions

/SECTIONS=SYMTAB

Sections of type SHT_SYMTAB will be analyzed and displayed. The data for this section is displayed as a symbol table.

/SECTIONS=STRTAB

Sections of type SHT_STRTAB will be analyzed and displayed. The data for this section is displayed as a string table.

/SECTIONS=RELOCATIONS

Sections of type SHT_RELA will be analyzed and displayed. The data for this section is displayed as table of relocation entries.

/SECTIONS=NOTE

Sections of type SHT_NOTE will be analyzed and displayed. The data for this section is displayed as a list of formatted OpenVMS note entries.

/SECTIONS=NOBITS

Sections of type SHT_NOBITS will be analyzed and displayed. There is no module data associated with sections of this type.

/SECTIONS=GROUP

•Sections of type SHT_GROUP will be analyzed and displayed. Sections of this type begin with a flagsword followed by a list of the section numbers of sections belonging to that group.

The sections belonging to the group will also be displayed.

/SECTIONS=TRACE

Sections of type SHT_VMS_TRACE will be analyzed and displayed. The data in these sections consists of OpenVMS traceback information.

/SECTIONS=LINKAGES

Sections of type SHT_VMS_LINKAGES will be analyzed and displayed. The data is displayed as a list of linkage descriptors.

/SECTIONS=SYMBOL_VECTOR

Sections of this type will only appear in image files. If present, it should point to the same data as the dynamic segment DT_VMS_SYMVEC tags.

/SECTIONS=EXTENSIONS

Sections of type SHT_IA64_EXT will be analyzed and displayed. The data is displayed in hexadecimal.

/SECTIONS=UNWIND

Sections of type SHT_IA64_UNWIND are analyzed and displayed. Each section of this type has an associated Unwind Information section of type SHT_PROGBITS. This associated section is also displayed.

/SEGMENTS Qualifier Keywords

/SEGMENTS [=(keyword1 [,...])]

•This qualifier is used to select individual program segments or program segments of a specified type to be displayed.

•Note: This qualifier and its keywords are used only to form an inclusion list of segments to be displayed. You cannot use a negative form of this qualifier (/NOSEGMENT) to form an exclusion list.

/SEGMENTS=HEADERS

The default keyword. ANALYZE will analyze and display the details of each segment contained in the segment header.

/SEGMENTS=ALL

The information for every program segment is analyzed and displayed. Note that this can generate a large amount of output.

/SEGMENTS=NUMBERS=(number [,…])

•This keyword will allow individual segments to be analyzed and displayed. The selected segments will have a detailed display of their header and their contents. An informational message is displayed for section numbers that do not exist in the module.

•One or more numeric values may be specified.

•Segment numbers may be specified in decimal, octal (using the %O prefix), or hexadecimal (using the %X prefix).

/SEGMENTS=NULL

Segments of type PT_NULL will be analyzed and displayed. No data will be displayed for segments of this type.

/SEGMENTS=LOAD

Segments of type PT_LOAD will be analyzed and displayed. If the segment header indicates this is an executable segment (PHDR$M_PF_X bit set in the segment header), the contents will be formatted as machine instructions; otherwise the contents will be formatted as hexadecimal data.

/SEGMENTS=CODE

ANALYZE will analyze and display all executable segments (PHDR$M_PF_X bit set in the segment header). Segment data will be displayed as machine instructions.

/SEGMENTS=DYNAMIC

The segment of type PT_DYNAMIC will be analyzed and displayed. There may be only one of these segments. On OpenVMS, this segment has a special meaning. Further detail on this qualifier appears below.

/SEGMENTS=EXTENSIONS

Segments of type IA_64_ARCHEXT will be analyzed and displayed.

/SELECT Qualifier Keywords

ANALYZE creates DCL symbols for all selectable information with the /SELECT qualifier. The symbol names consist of the prefix ANALYZE$ and a descriptive name of the information they hold.

The symbol value is the selected information, usually printed to SYS$OUTPUT. Effectively all of the printed information is duplicated in the symbols. For unselected information, the corresponding symbols will contain the null string.

/SELECT= ARCHITECTURE

Writes the architecture information into the DCL symbol ANALYZE$ARCHITECTURE.

/SELECT= BUILD_IDENTIFICATION

Writes build identification information into the DCL symbol ANALYZE$BUILD_IDENTIFICATION.

/SELECT= FILE_TYPE

Writes file-type information into the DCL symbol ANALYZE$FILE_TYPE.

/SELECT= IDENTIFICATION [=IMAGE]

Writes the image identification information into the DCL symbol ANALYZE$IDENTIFICATION.

/SELECT=IDENTIFICATION = LINKER

Writes the linker indentification information into the DCL symbol ANALYZE$LINKER_IDENTIFICATION.

/SELECT=IMAGE_TYPE

Writes image-type information into the DCL symbol ANALYZE$IMAGE_TYPE.

/SELECT=LINK_TIME

Writes link-time information into the DCL symbol ANALYZE$LINK_TIME.

/SELECT=NAME

Writes image-name information into the DCL symbol ANALYZE$NAME.

Note: The Analyze utility can work on several files. Because there is only one set of DCL symbols, the symbols contain information only from the last analyzed file. When an error occurs, symbol values are undefined. Check for ANALYZE errors first, then use the symbols.

Examples:

```
$ ANALYZE/IMAGE/SELECT=(ARCHITECTURE, IDENT, NAME)   HELLO [1]
USER:[JOE]HELLO.EXE;1
OpenVMS  IA64
"V1.0"
"HELLO"
$
$ SHOW SYMBOL ANALYZES$*
   ANALYZE$ARCHITECTURE = "OpenVMS IA64"
   ANAYLZE$BUILD_IDENTIFICATION = ""
   ANALYZE$FILE_TYPE = ""
   ANALYZE$IDENTIFICATION = ""V1.0""
   ANAYLZE$IMAGE_TYPE = ""
   ANALYZE$LINKER_IDENTIFICATION = ""
   ANAYLZE$LINK_TIME = ""
   ANALYZE$NAME = ""HELLO""
$
$ ANALYZE/IMAGE/SELECT=(IDENT=(IMAGE,LINKER), IMAGE,LINK)  HELLO  [2]
USER:[JOE]HELLO.EXE;1
"V1.0"
"Linker I01-54"
Executable
 7-NOV-2003  11:47:08.10
$
$ SHOW SYMBOL ANALYZE$*
   ANALYZE$ARCHITECTURE = ""
   ANALYZE$BUILD_IDENTIFICATION = ""
   ANALYZE$FILE_TYPE = ""
   ANALYZE$IDENTIFICATION = ""V1.0""
   ANALYZE$IMAGE_TYPE = "Executable"
   ANALYZE$LINKER_IDENTIFICATION = ""Linker I01-54""
      ANALYZE$LINK_TIME = " 7-NOV-2003  11:47:08.10 "
   ANALYZE$NAME = ""
$
$ ANALYZE/IMAGE/SELECT=FILE HELLO.*  [3]
USER:[JOE]HELLO.C;1
%ANALUZE=E-ILLFIL,  Illegal file format encountered
USER:[JOE]HELLO.EXE;1
Image
USER:[JOE]HELLO.MAP;1
%ANALYZE-E-ILLFIL, Illegal file format encountered
USER:[JOE]HELLO.OBJ;1
Object
$
$ SHOW SYMBOL ANALYZE$*
   ANALYZE$ARCHITECTURE = ""
```

```
   ANALYZE$BUILD_IDENTIFICATION = ""
   ANALYZE$FILE_TYPE = "Object"
   ANALYZE$IDENTIFICATION = ""
   ANALYZE$IMAGE_TYPE = ""
   ANALYZE$LINKER_IDENTIFICATION = ""
   ANALYZE$LINK_TIME = ""
   ANALYZE$NAME = ""
$
```

Callouts:

1. Only the selected information can be found in the DCL symbols. The information in the symbols is identical to what is printed to SYS$OUTPUT. That is, if quoted strings are printed, there are quote strings in the symbol.
2. If the new linker identification is selected, /IDENT with a keyword list is necessary.
3. When using wildcards, errors in the analyzed file (for example, illegal file format errors) do not terminate ANALYZE. Only the information from the last analyzed file can be found in the DCL symbols.

/FLAGVALUES Qualifier Keywords

/FLAGVALUES=[ON] [OFF] [ALL]

•Several fields in an ELF module represent bit flags. Where possible, these bitflag values are examined and displayed individually.

•By default, only the flag values that are set to 1 (that is, ON) are displayed.

/FLAGVALUES=ON

All flags whose value is 1 are displayed. The keyword ON is used when displaying them.

/FLAGVALUES=OFF

All flags whose value is 0 are displayed. The keyword OFF is used when displaying them.

/FLAGVALUES=ALL

All flag values are displayed. The keywords ON and OFF are used to indicate the value of each specific flag bit.

The Dynamic Segment (/SEGMENTS=DYNAMIC)

On OpenVMS, the dynamic segment is used to hold much of the information the OpenVMS image activator needs to load and run the image. This is a normal ELF program segment and the information the image activator needs is either in the dynamic segment or reached through references in the dynamic segment. There will be only one dynamic segment in an image.

The first portion of the dynamic segment analysis is called the DYNAMIC SEGMENT TABLE and is structured as follows:

- Each element in the table contains a tag name indicating how this element should be interpreted and a tag value that is interpreted.
- This table is always displayed (if present) as part of the summary information appearing at the start of the analysis.

- In addition, the Shareable Image List and the System Version Array (if present), two of the elements referred to by the DYNAMIC SEGMENT TABLE, are also always displayed.
- The remainder of the entries in the DYNAMIC SEGMENT TABLE either contain the associated information in the tag value field (for example, DT_VMS_IDENT) or refer to data located elsewhere in the image file.
- Other tags are used in combination to describe one or more objects. For example, the three tags DT_VMS_SYMVEC_CNT, DT_VMS_SYMVEC_OFFSET, and DT_VMS_SYMVEC_SEG refer to the image's symbol vector.
- If the dynamic segment is selected for display, all information referred to by the dynamic segment table is displayed regardless of whether this information is located in the dynamic segment or is located elsewhere in the image file.

NOTE ANALYZE does not support a method to dump only the contents of the dynamic segment itself. If this is desired, the VMS DUMP utility may be used.

Qualifiers Ignored by I64 Modules

•/DBG

•/EOM

•/GSD

•/LNK

•/MHD

•/TBT

•/TIR

•/FIXUP_SECTION

•/GST

•/HEADER

•/PATCH_TEXT

Qualifiers Ignored for OpenVMS VAX and Alpha

•/SECTIONS

•/SEGMENTS

•/FLAGVALUES

Features Not Yet Implemented

- Formatting of debug (DWARF) sections.
- Formatting of translated image relocations and signatures.
- Formatting to function descriptors that are more that two quadwords in length.

LIBRARIAN Utility Release Notes

This section provides information about the Librarian utility

Librarian Usage Summary

You can use the DCL LIBRARY command (or Librarian LBR routines) to create libraries such as I64 (ELF) Object library, I64 (ELF) Shareable image library, Macro library, Help library, and Text library, maintain the modules in a library, or display information about a library and its modules. The Librarian utility provides the same features provided by the Alpha Librarian except for the following changes and restrictions.

Table 3-2 Libraries Created by the Librarian Utility by Platform

OpenVMS VAX	OpenVMS Alpha	OpenVMS I64
VAX object	Alpha object	I64 object
VAX shareable image	Alpha shareable image	I64 shareable image
Alpha object	VAX object	Macro
Alpha shareable image	VAX shareable image	Text
Macro	Macro	Help
Text	Text	
Help	Help	

NOTE The performance of the I64 Librarian decreases slightly compared with the Alpha Librarian. This performance limitation will be eliminated in a future release.

Changes to LIBRARY Command

The following sections list the changes to the LIBRARY command.

Librarian Defaults to I64 Architecture

The default architecture for the Librarian is I64. There is no architecture switch for I64 libraries. The Librarian works with OpenVMS ELF object and image libraries when used with the following qualifiers:

/OBJECT — Work on OpenVMS ELF object libraries (default)
/SHARE — Work on OpenVMS ELF shareable image libraries
/CREATE — Create an OpenVMS ELF library of an object or shareable image type, depending on whether the /OBJECT or /SHARE qualifier is specified.

The default library type created is an object library if no /OBJECT or /SHARE qualifiers is specified.

No Support for /ALPHA and /VAX Qualifiers

The /ALPHA and /VAX qualifiers are not supported in the Librarian utility. The Librarian utility works on the following library types only:

- ELF OBJECT
- ELF SHAREABLE IMAGE
- HELP
- MACRO
- TEXT

Restrictions

These are the restrictions for the Librarian utility on an I64 system:

- The maximum length of symbols in OpenVMS ELF object and shareable image libraries is 128 characters.
- The /CROSS_REFERENCE qualifier for OpenVMS ELF object library is not yet implemented.

Changes to Librarian LBR Routines

Two new library types for the LBR$OPEN routine have been added:

LBR$C_TYP_ELFOBJ (9) — Represents an ELF object library
LBR$C_TYP_ELFSHSTB (10) — Represents an ELF shareable image library

In addition, the following library types for the LBR$OPEN routines are not supported in the Librarian. You cannot use the library types to create or open OpenVMS Alpha or OpenVMS VAX object and shareable image libraries:

LBR$C_TYP_OBJ (1) — Represents a VAX object library
LBR$C_TYP_SHSTB (5) — Represents a VAX shareable image library
LBR$C_TYP_EOBJ (7) — Represents an Alpha ohject library
LBR$C_TYP_ESHSTB (8) — Represents an Alpha shareable image library

Library Format Changed Due to New UNIX-Style Weak Symbols

Due to the requirements of the Intel® C++ compiler, the library format has been expanded to accommodate new UNIX-style weak symbols. Multiple modules matching key names of new UNIX-style weak symbols can now exist in the same library. The Librarian ignores the VMS-style weak symbol definitions as it has in the past.

UNIX-style weak symbol definitions behave in the same manner as weak transfer addresses on OpenVMS; that is, their definitions are tentative. If a definition of a stronger binding type is not seen during a link operation, the tentative definition is designated as the definitive definition.

New ELF Type for Weak Symbols

A new Executable and Linkable Format (ELF) type was generated to distinguish between the two types of weak symbol definitions.

For modules with ABI versions equal to 2 (the most common version used by compilers):

- Type STB_WEAK represents the UNIX-style weak symbol (formerly, the VMS-style weak symbol definition for ABI Version 1 ELF format).
- Type STB_VMS_WEAK represents the VMS-style weak symbol definition.

The Librarian supports both the ELF ABI version 1 and 2 of the object and image file formats within the same library.

NOTE The new library format (Version 4.0) applies only to ELF object and shareable image libraries. Other libraries will remain at Version 3.0 format. Applications that reference the library by way of the currently-defined library services interface should not see any change in behavior.

Version 4.0 Library Index Format

HP recommends using the new Version 4.0 libraries.

With the new library index format (Version 4.0), the Library Services opens Version 3.0 libraries in read-only mode. You cannot modify the library with the Library Services or with the Librarian. Instead, you can convert your Version 3.0 libraries to Version 4.0 libraries using the following Library command:

```
$ library/compress library-name
```

Once your library is in Version 4.0 format, you will not be able to access the library with older versions of the Library Services or Librarian. Doing so will cause the Library Services to return the LBR$_UNSUPLVL status, and will cause the Librarian to display the following message:

```
%LIBRAR-F-OPENIN, error opening library-name as input
-LBR-E-UNSUPLVL, unsupported library format level
```

Current Library Limitation with Regard to Weak and Group Symbols

Library symbol entries are associated with the module within which they were defined. On an OpenVMS I64 system, more than one module can define a UNIX-style weak or Group symbol. As such, the Librarian must keep an ordered list of defining modules. This list allows a previous association to be restored should the higher precedence association be removed. (See the section, Precedence Ordering Rules, for more information.)

Currently, the ordered list of defining modules and the associated restoration process is not implemented. As a result, associations that would otherwise have been saved are lost when overwritten. If the top-most association of a symbol is removed, no previous association can be restored and the entry is removed from the symbol name index. The list of associations will be implemented in a future release of the Library Services.

For the current implementation, HP suggests that you perform only insert operations into the library for modules that contain UNIX-style weak definitions. Should you decide to remove or replace modules in the library, you need to rebuild the library to make sure that the association of UNIX-style weak definitions is correct.

New Group-Section Symbols

Symbols may be relative to sections contained in an ELF entity called a *group*. These groups, and the symbols associated with them, behave in a similar fashion as the new UNIX-style weak symbol definitions; that is, they are tentative definitions. The library must now allow multiple symbol definitions in the library's symbol name index.

Precedence Ordering Rules

The following list describes the symbol types for the combinations of GROUP and UNIX-style WEAK attributes in order of their precedence from highest to lowest. Within these symbol types, the ordering of symbol associations is by chronological time of insertion, from earliest to latest.

1. Non-Group Global symbols – Symbols that are not members of an ELF group and are not weak definitions. These symbols have the highest precedence. In the symbol name table, they take the place of any lower-precedence symbol definition that may already exist. There may be only one definition of this type. If the symbol of this type already exists in the library, a subsequent symbol definition of this type returns an error.

2. Group Global symbols – Symbols that belong to an ELF section associated with a group. For the new Library Services, a symbol of this type may have its association overwritten by a Non-Group Global symbol but not by any other symbol type. (In the current implementation, if the symbol is overwritten, the association of the symbol to the owning library module is lost and cannot be restored. This association will be saved and restored in a future version of the Library Services.)
3. UNIX-Style Weak symbols – Symbols that have a UNIX-style weak binding. The new Library Services handles these symbols similar to the way it handles Group Global symbols. A future release of the Library Services will differentiate between Group Global symbols and UNIX-style Weak symbols with regard to precedence.
4. Group Weak symbols – Symbols that behave as combined Group Global and UNIX-Style Weak attributes. Placement of these symbols in the library name table is also similar to Group Global symbols.

CHECKSUM Utility for I64 Systems

The Checksum utility calculates checksums for an OpenVMS file. It performs an image or object checksum following the Executable and Linkable Format ELF64 structure in the file or a file checksum for all bytes within the file, regardless of its internal structure.

The Checksum utility uses a CRC-32 algorithm for all checksums. The CRC is known as AUTODIN II, Ethernet, or FDDI CRC and is documented with the VAX CRC instruction. The same algorithm is used by popular compression tools like PKZIP. That is, a file checksum in a ZIP file can be compared with the file checksum obtained by checksum.

The image or object checksum follows the ELF64-bit data structures, as they are used for OpenVMS I64 object and image files. For these checksums, only the invariant data is used for the calculation. Variant data, such as timestamps and versions, are excluded from the calculation. This allows for comparisons between results from different versions of the same language processor.

The Checksum utility is available as a native utility on OpenVMS I64 systems. It accepts several qualifiers that influence the type of checksum and the type of information shown to the user.

In general, the Checksum utility for OpenVMS I64 behaves like the comparable OpenVMS VAX and OpenVMS Alpha tools in that the utility accepts the same set of qualifiers and prints similar information.

Using the Checksum Utility

The CHECKSUM command invokes a utility to calculate one or more CRC-based checksums for OpenVMS files.

Format:

CHECKSUM *filespec*

Parameters:

filespec - Specifies the name of an existing file to be displayed. The asterisk (*) and the percent sign (%) wildcard characters are allowed.

Checksum Qualifiers:

The /FILE, /IMAGE and /OBJECT qualifiers are file type qualifiers. They imply a default type (.DAT, .EXE and .OBJ, respectively) and specify the kind of information shown. The default qualifier is /FILE, which implies a default file type, .DAT. The information displayed is the file checksum.

NOTE The /VAX and /ALPHA qualifiers do not function in the Checksum utility for I64 systems.

/FILE Qualifier

Format:

/FILE (D)

Description:

A checksum of all bytes of a file is calculated, regardless of the internal structure of the file.

The full filename of the specified input file is printed as well as the file checksum. The checksum value is printed in hexadecimal notation. Additionally, the unsigned decimal checksum value is saved in the DCL symbol `CHECKSUM$CHECKSUM`. The qualifier assumes a default file type of .DAT. The qualifier assumes the default value DATA for the SHOW qualifier.

/IMAGE Qualifier

Format:

/IMAGE

Description:

A checksum of all image bytes is calculated. Checksum uses the ELF64 structure of the OpenVMS image format to calculate the checksum from portions of the image that do not vary from subsequent link operations, such as the image link time or linker version string.

The full filename of the specified input file is printed as well as the image checksum. The checksum value is printed in hexadecimal notation. Additionally, the unsigned decimal checksum value is saved in the DCL symbol `CHECKSUM$CHECKSUM`. The qualifier assumes a default file type of .EXE. The qualifier also assumes the default value STRUCTURE for the SHOW qualifier.

/OBJECT Qualifier

Format:

/OBJECT

Description:

A checksum of all object bytes is calculated. Checksum uses the ELF64 structure of the OpenVMS object format to calculate the checksum from portions of the object file that do not vary from subsequent compilations, such as the language processor version and file creation date.

The full filename of the specified input file is printed as well as the object checksum. The checksum value is printed in hexadecimal notation. Additionally, the unsigned decimal checksum value is saved in the DCL symbol `CHECKSUM$CHECKSUM`. The qualifier assumes a default file type of .OBJ. The qualifier also assumes the default value STRUCTURE for the SHOW qualifier.

/OUTPUT Qualifier

Format:

/OUTPUT = [*filespec*]
/NOOUTPUT

Description:

Optionally directs the text output to a file. If you specify the /OUTPUT=*filespec* qualifier, the output is sent to the specified file, rather than to the current output device, SYS$OUTPUT. If you do not enter the qualifier, or if you enter the /OUTPUT qualifier without a file specification, the output is sent to SYS$OUTPUT.

If you enter the /NOOUTPUT qualifier, output is suppressed, however, the unsigned decimal checksum value is still saved in the DCL symbol CHECKSUM$CHECKSUM.

/SHOW Qualifier

Format:

/SHOW=*option* [,...]

Description:

Controls which checksum and which additional information is printed to the output device. Possible options are:

HEADERS – Print checksums of all ELF64 headers. This option is set by default for /IMAGE and /OBJECT.
SEGMENTS – Print checksums of all ELF64 program segments. This option is set by default for /IMAGE.
SECTIONS – Print checksums of all ELF64 sections. This option is set by default for /OBJECT.
EXCLUDED – Format the data excluded from the image or object checksums.
DATA – Print the file checksum. This is the default for /FILE.
ALL – Print all of the appropriate information depending on the file type.

For a file checksum, only the DATA keyword is accepted. For an image checksum, all keywords are accepted. For an object checksum, the keyword SEGMENT is not accepted.

Checksum Utility Examples

```
$ CHECKSUM X
File USER:[JOE]X.DAT;1
File checksum: %X0403C414
```

The file checksum for X.DAT is calculated. The default /FILE is used, which also implies a default extension.

```
$ CHECKSUM HELLO.EXE
File USER:[JOE]HELLO.EXE;9
File checksum: %XD7006308
```

The file checksum for the image HELLO.EXE is calculated. The default /FILE is used so that it is an image, the ELF structure is not checksummed.

```
$ CHECKSUM /IMAGE USER:[JOE]HELLO
File USER:[JOE]HELLO.EXE;9
Checksum program segment 0: %X0DD49EF0
Checksum program segment 1: %X13B51DFF
```

```
Checksum program segment 2: %X4E4CD3A9
Checksum program segment 3: %XF181EA61
Checksum dynamic segment %X839CCB27

Elf header checksum: %X1B43E818
Elf program header checksum: %X0956F6B3
Elf section header checksum: %X2F35D964
Elf (object/image) checksum: %XA637981D
```

The image checksum for the ELF image, HELLO.EXE, is calculated. The file type /IMAGE is requested, default file type is implied, and the default output is shown:

```
$ CHECKSUM /IMAGE USER:[JOE]HELLO.EXE/NOOUTPUT
$ SH SYMB CHECKSUM$CHECKSUM
CHECKSUM$CHECKSUM = "2788661277"
```

The image checksum for the ELF image HELLO.EXE is calculated. The file type /IMAGE is requested and the output is suppressed.

```
$ CHECKSUM /IMAGE USER:[JOE]HELLO.EXE/SHOW=ALL
File USER:[JOE]HELLO.EXE;9
Checksum program segment 0: %X0DD49EF0
Checksum program segment 1: %X13B51DFF
Checksum program segment 2: %X4E4CD3A9
Checksum program segment 3: %XF181EA61
Dynamic segment, match control: ISD$K_MATALL
                 major id: %X0001
                 minor id: %X0000609C
Dynamic segment, linktime: 28-DEC-2003 12:38:20.21
Checksum dynamic segment %X839CCB27
Checksum section 1: %X0F9601F5
Checksum section 2: %X68D12696
Checksum section 3: %X964C4840
Section 4 excluded: note section
Note entry, owner: 'IPF/VMS'
            imgnam: 'HELLO'
Note entry, owner: 'IPF/VMS'
            imgid: 'V1.0'
Note entry, owner: 'IPF/VMS'
            linktime: 28-DEC-2003 12:38:20.21
Note entry, owner: 'IPF/VMS'
            linkid: 'Linker I01-57'
Checksum section 5: %XBF3AE511
Checksum section 6: %XA8FF2D5A
Elf header checksum: %X1B43E818
Elf program header checksum: %X0956F6B3
Elf section header checksum: %X2F35D964
Elf (object/image) checksum: %XA637981D
File checksum: %XD7006308
```

The image and file checksum for the ELF image HELLO.EXE is calculated. The file type /IMAGE is requested, all possible output is displayed.

```
$ CHECKSUM /IMAGE USER:[USER]HELLO.EXE/SHOW=EXCLUDED
File USER:[JOE]HELLO.EXE;9
Dynamic segment, match control: ISD$K_MATALL
                 major id: %X0001
                 minor id: %X0000609C
Dynamic segment, linktime: 28-DEC-2003 12:38:20.21
Note entry, owner: 'IPF/VMS'
            imgnam: 'HELLO'
```

```
Note entry, owner: 'IPF/VMS'
            imgid: 'V1.0'
Note entry, owner: 'IPF/VMS'
            linktime: 28-MAR-2003 12:38:20.21
Note entry, owner: 'IPF/VMS'
            linkid: 'Linker I01-57'
Elf (object/image) checksum: %XA637981D
```

The image checksum for the ELF image HELLO.EXE is calculated. The file type /IMAGE is requested, the excluded data is formatted.

```
$ CHECKSUM /OBJECT USER:[JOE]CORRUPT_NOTE.OBJ/SHOW=ALL
File USER:[JOE]CORRUPT_NOTE.OBJ;3
Checksum section 1: %XFAE35D2B
Section 2 excluded: note section
Note entry, owner: 'IPF/VMS'
            module date: '20-DEC-2003 10:16'
            patch date: '20-DEC-2003 10:16'
            module name: 'SYSTEM_ROUTINES_MASK'
            module version: 'HB-4711'
Note entry, owner: 'IPF/VMS'
            unknown type: %X000000000000000B
            descriptor: %X494D41432054312E302D343700
Section 3 has filesize 0
Section 4 has filesize 0
Checksum section 5: %XCC9326D6
Checksum section 6: %X0EFAC4C4
Elf header checksum: %X696C98BA
Elf section header checksum: %X2A540BB4
Checksum section 1: %X2097FD20
Section 2 excluded: note section
Note entry, owner: 'IPF/VMS'
            module date: '20-DEC-2003 10:16'
            patch date: '20-DEC-2003 10:16'
            module name: 'BASE_DATA'
            module version: 'V1'
Note entry, owner: 'IPF/VMS'
            language processor: 'IMAC T1.0-47'
Section 3 has filesize 0
Checksum section 4: %XCE771B9E
Checksum section 5: %X172C3661
Checksum section 6: %XA66B80DC
Checksum section 7: %X21F1D74C
Elf header checksum: %XC82E68C1
Elf section header checksum: %X049DBD44
Elf (object/image) checksum: %XD2620510
File checksum: %X0D87EFCF
```

The object checksum for the ELF object CORRUPT_NOTE.OBJ is calculated. The file type /OBJECT is requested, and all output is displayed.

Note that there are two modules in this file, therefore, two ELF header and section header checksums are shown. However, notice that there is only one ELF (object) and one file checksum. Note also that the second note entry is corrupt. Because it cannot be formatted, its values are printed in hexadecimal.

```
$ CHECKSUM /IMAGE USER:[JOE]HELLO.*/SHOW=DATA
File USER:[JOE]HELLO.C;1
%CKSM-W-NOTELF, not an Elf object/image file
File checksum: %XC96E4185
File USER:[JOE]HELLO.DSF;5
```

```
Elf (object/image) checksum: %X01A0F178
File checksum: %XFF000F7D
File USER:[JOE]HELLO.EXE;9
Elf (object/image) checksum: %XA637981D
File checksum: %XD7006308
File USER:[JOE]HELLO.LIS;1
%CKSM-W-NOTELF, not an Elf object/image file
File checksum: %X00DD6C58
File USER:[JOE]HELLO.OBJ;1
Elf (object/image) checksum: %XEF0FEAE8
File checksum: %XC5686048
```

The image and file checksums for the files matching the specification HELLO.* are calculated. The file type /IMAGE is requested, and the file checksum /SHOW=DATA is displayed. Note that some files are not ELF image files: HELLO.C, HELLO.DSF, HELLO.LIS and HELLO.OBJ. However, some have an ELF structure, which is used to calculate the Elf (object/image) checksum: HELLO.DSF and HELLO.OBJ. All the others are flagged as non-ELF files, and only the requested file checksum is displayed.

Differences Between the OpenVMS I64 Checksum Utility and Other Versions on OpenVMS.

The Checksum utilities for OpenVMS VAX and OpenVMS Alpha are no longer supported or fully documented. Partial documentation is in the *VMSINSTAL Developer's Guide*.

The following differences exist between the Checksum utility on OpenVMS I64 systems and OpenVMS VAX and Alpha systems:

- The VAX and Alpha checksums are not based on a CRC. Therefore, the checksums usually are different, even for data.
- Checksum for Alpha accepts the /ALPHA and /VAX qualifiers. It checksums both Alpha and VAX images. There are no differences for data files.
- Checksum for Alpha does not always print the file name. For file checksums, the file name is not printed, however, it is for image files. The Checksum for I64 systems always prints the file name.
- Checksum for Alpha does not always print the checksum. The DCL symbol CHECKSUM$CHECKSUM is set for file checksums only. The Checksum for I64 systems always prints the checksum.
- Because there are different image structures, the names for the checksums differ.
- Checksum for Alpha prints section numbers as BLISS constant: %D'1'. The Checksum utility for I64 systems prints section and segment numbers as decimals.
- Checksum for Alpha prints the checksums as BLISS constant: %X'6C5404CB'. The Checksum for I64 systems prints DCL-style hexdecimal numbers %XC96E4185.
- The /OBJECT qualifier is new for the Checksum utility for I64 systems.
- The /SHOW qualifier is new for the Checksum utility for I64 systems. This qualifier also prints the data that is not used for the image or object checksum.
- Checksum for Alpha with /NOOUTPUT still writes a file. This is fixed in the Checksum for I64 systems.
- Checksum for Alpha for a non-existing file gives a parse error. The Checksum utility for I64 systems uses its own messagses.
- Checksum for Alpha with /IMAGE and a wildcard file specification aborts if there is a non-Alpha image. The Checksum utility for I64 systems prints a message and continues with the next file.

Threads Libraries Notes

The user-mode threads facilities for OpenVMS I64 have some temporary restrictions in this kit.

1. Use of multiple kernel threads in one process is not supported.

2. Exception handling on I64 systems requires considerably more stack space than it does on Alpha. When porting an application from OpenVMS Alpha, if a thread that uses exception handling does not have a generous amount of unused stack space, then it is possible for the thread to experience a stack overflow during exception handling on I64. Usually this will appear as an improperly handled ACCVIO that is associated with one of the following operations:

- RAISE
- pthread_cancel
- pthread_exit
- Any time that a thread has an active TRY or pthread_cleanup_push block, and an OpenVMS condition is signaled (for example, via LIB$SIGNAL or LIB$STOP, or as a hardware exception).

If you see such a problem, try increasing the size of the stack allocated for the thread by a small number of pages. We recommend initially increasing the stack by 24 KB.

The default stack size has increased by 24 KB to try and address the increased stack usage on I64 systems. If your application creates a large number of threads (using the default size), the application may run out of memory resources. This could require the user to increase process quotas, or could require application changes to reduce the number of threads which exist simultaneously.

HP Fortran T8.0 for OpenVMS Industry Standard 64 Integrity Servers

The OpenVMS I64 Fortran compiler is a port of HP Fortran 90 for OpenVMS Alpha. It runs on OpenVMS I64 systems and produces objects for OpenVMS I64 systems. The objects are l inked using the standard linker on OpenVMS I64. This compiler requires OpenVMS Industry Standard 64 Evaluation Release Version 8.1 for Integrity Servers. Some highlights for this release follow.

HP Fortran for OpenVMS I64 systems features the same command-line options and language features as HP Fortran 90 for OpenVMS Alpha systems with a few exceptions, specifically:

- Floating-point arithmetic

  The white paper *OpenVMS Floating-Point Arithmetic on the Intel® Itanium® Architecture* is essential reading. This document can be found on the following web site:

  http://www.hp.com/products1/evolution/alpha_retaintrust/download/i64-floating-pt-wp.pdf

  — IEEE is the default floating-point datatype (that is, the default is /FLOAT=IEEE_FLOAT.).

  — The /IEEE_MODE qualifier defaults to /IEEE_MODE=DENORM_RESULTS

  — Users must pick one /FLOAT value and one /IEEE_MODE value and keep it for the whole of their application.

- Only the FS90 compiler is supported. The F77 compiler, previously invoked with the /OLD_F77 qualifier, is not available. Development is currently underway to provide some of the functionality contained in the Alpha F77 compiler that is not available on the I64 and Alpha F90 compilers, including FDML and CDD support. See the HP Fortran T8.0 for OpenVMS Industry Standard 64 for Integrity Servers Release Notes for details.
- Alpha values for the /ARCH and /TUNE qualifiers are accepted on the compiler invocation command, for compile-and-go compatibility. An informational message is displayed saying that they are ignored.

For more information about this release, including installation instructions, read the Fortran T8.01 product release notes, which constitute the only I64-specific documentation available for this beta release. To extract the release notes as a text file, enter the following:

```
$ set def directory-containing-Fortran-PCSI-kit
$ product extract release_notes fortran

The following product has been selected:
        HP I64VMS FORTRAN T8.01               Layered Product

Do you want to continue?  [YES]   <enter RETURN here>

Portion done:  0%...100%
%PCSI-I-CREFIL, created disk:[directory]FORTRAN.RELEASE_NOTES;1
```

To extract the release notes in PostScript form, enter the following:

```
$ set def directory-containing-Fortran-PSCI-kit
$ product extract file fortran/select=fortran_release_notes.ps

The following product has been selected:
        HP I64VMS FORTRAN T8.0-1       Layered Product

Do you want to continue?  [YES]   <enter RETURN here>

Portion done:  0...100%
$
```

HP BLISS for OpenVMS Release Notes

The following sections list release notes for the BLISS compiler on OpenVMS I64 systems.

Possible BLISS Compiler Warnings

Quadword alignment has been added to the BLISS macros ($xxx_DECL) that can be used to allocate RMS user structures (for example, FAB, RAB). Alignment faults are costly to performance -- even more so as processors get faster. By implementing the alignment directly in the macros, a number of OpenVMS utilities and user applications written in BLISS that use these macros see improved performance.

The specific names of the macros are: $FAB_DECL, $NAM_DECL, $NAML_DECL, $RAB_DECL, $RAB64_DECL, $XABALL_DECL, $XABDAT_DECL, $XABFHC_DECL, $XABITM_DECL, $XABJNL_DECL, $XABKEY_DECL, $XABPRO_DECL, $XABRDT_DECL, $XABRU_DECL, $XABTRM_DECL, and $XABSUM_DECL.

The alignment added in the RMS macros may result in compiler warnings of conflicting alignment. Programs with compiler warnings should link and execute correctly. However, the minor source changes to eliminate the warnings are recommended.

If you use any of these macros in a BLISS application and the declaration includes the ALIGN attribute, then the BLISS compiler will issue a "conflicting or multiply specified attribute" warning. For example, the warning would be issued for the following declaration: FAB: $FAB_DECL ALIGN(2). It would also be issued even if quadword alignment (ALIGN(3)) were specified. Any explicit ALIGN attributes associated with these macros should be removed.

In addition, if any of these allocations is included in a PSECT that contains an explicit alignment that is in conflict with ALIGN(3) (that is, is lower than ALIGN(3)), then the BLISS compiler will issue an "align request negative or exceeds that of psect" warning. For example, the warning would be issued for the following declaration:

```
PSECT OWN = $OWN$ (..., ALIGN(2), ...)

OWN

    FAB = $FAB_DECL, ...
```

If warnings on psect alignment are seen in recompiling a BLISS application, the correction would be to increase the alignment of the PSECT to ALIGN(3) (or higher). In rare cases, applications may have assumptions on adjacency of data between psects. Those assumptions could be broken in making this change. Therefore, code should be checked for such assumptions and necessary corrections made.

While a number of OpenVMS utilities are written in BLISS, only a few warnings were generated in a full OpenVMS build. Changes in OpenVMS to eliminate warnings did not require other changes to correct assumptions. Based on this, our expectation is that few user applications will require modification.

BLISS Precompiled Library Files must be Compiled Manually

The BLISS precompiled library files must be compiled manually after installing the BLISS compiler. These library files are built using the following DCL commands from the SYSTEM account:

```
$ bliss/i32/lib=sys$common:[syslib]starlet.132 sys$library:starlet.req
$ bliss/i32/lib=sys$common:[syslib]lib.132 sys$library:lib.req
$ bliss/i64/lib=sys$common:[syslib]starlet.164 sys$library:starlet.r64
$ bliss/i64/lib=sys$common:[syslib]lib.164 sys$library:lib.r64
```

The BLISS complilation generates several information messages, which you can safely ignore.

OpenVMS I64 Debugger Release Notes

The following sections describe the capabilities, resolved problems, limitations and known problems of the OpenVMS I64 Debugger. A general knowledge of the debugger capabilities on OpenVMS Alpha is assumed.

Debugger Capabilities in This Release

This section lists the supported languages, commands, and capabilities of the debugger.

Architecture Support

The OpenVMS I64 Debugger supports the following Itanium hardware registers:

- General registers R0 through R127.
- Floating registers F0 through F127.
- Branch registers B0 through B7.

- A 64-bit predicate value named PRED, representing predicate registers P0 through P63.
- Application registers: AR16 (RSC), AR17 (BSP), AR18 (BSPSTORE), AR19 (RNAT), AR25 (CSD), AR26 (SSD), AR32 (CCV), AR36 (UNAT), AR64 (PFS), AR65 (LC), AR66 (EC).
- A program counter named PC, synthesized from the hardware IP register and the ri field of the PSR register.
- Miscellaneous registers: CFM (current frame marker), UM (user mask), PSP (previous stack pointer), and IIPA (previously executed bundle address).

Language Support

The OpenVMS I64 Debugger supports programs written in the following languages:

- BLISS
- C
- C++ (limited)
- COBOL (limited)
- MACRO-32
- Fortran
- Intel® assembler (IAS)

Support for C++ and COBOL in this release is limited. These limitations are described in the section entitled, Limitations, Restrictions, and Known Problems.

Functional Areas and Commands

The following functional areas and commands are available:

- DECwindows Graphical User Interface (source display only)
- Screen mode: source display, instruction display, output display
- Instruction decoding
- Breakpoints and tracepoints
- Watchpoints (non-static)
- Step (all forms)
- $ DEBUG/KEEP (kept debugger configuration)
- $ RUN/DEBUG (normal debugger configuration)

Resolved Problems

The following debugger problems in OpenVMS I64 Version 8.0 have been fixed in OpenVMS I64 Version 8.1:

- Multi-threaded programs are now supported.
- The CALL command is now supported.
- Debugger commands no longer intermittently fail with an infinitely repeating DEBUG-E-CMDFAILED error message.
- Symbolic debugging of code in shareable images is now supported.
- Debugger commands no longer intermittently fail with the error DEBUG-E-RPCINVDSC.

- The DEBUG-I-INCMPSTACK message no longer appears when your program runs to completion.
- The debugger now symbolizes all possible static data locations.
- Values in floating-point registers are now displayed as IEEE T-floating values.
- Examines and deposits of floating-point variables now work.
- Problems with duplicate symbols resulting in the DEBUG-I-NOUNIQ error have been resolved.
- The EXAMINE/ASCII command now works.

Limitations, Restrictions, and Known Problems

This section describes general limitations, restrictions, and known problems of the OpenVMS I64 Debugger for this release.

Functionality Not Yet Ported

The following capabilities have not yet been ported to the OpenVMS Debugger for I64 systems:

- DECwindows Graphical User Interface: instruction view, register view, source browser
- Heap analyzer
- Screen mode register view

Known Problems

The following problems have been identified, along with user workarounds:

Problem: Entering either the SHOW TASK or the SHOW THREAD command in nonmulti-threading program environments results in a Debugger access violation following by a debugger internal error.

Workaround: Do not use these commands unless you are debugging a threaded program.

Problem: Typing Ctrl/C to interrupt an executing Debugger command results in an improperly handled condition and register dump, following by SYSTEM-F-BADSTACK.

Workaround: None.

Problem: A watchpoint on a floating-point register – or a variable in a floating-point register – watches only the lower 64-bits of the value, not all 128 bits. Although the watchpoint does track changes to the register, the Debugger will sometimes report an illegal floating-point number instead of the value in the register.

Workaround: Examine the register to see the correct value.

Problem: The CONNECT command may result in an improperly handled condition and register dump, followed by a SYSTEM-F-BADSTACK message.

Workaround: Start the program under debug control.

Problem: Attempting to use the debugger's DECwindows Instruction View can cause either a Debugger hang or a system crash.

Workaround: Do not activate the DECwindows Instruction View.

Problem: Pressing the STOP button on the DECwindows interface causes the debugger to crash with an improperly handled condition and register dump, followed by a SYSTEM-F-BADSTACK message.

Workaround: None

BLISS Language Issues

Problem: BLISS structure references containing literal values, for example, [0, 0, 32, 0], are not supported.

Workaround: Define the BLISS structure using field sets and refer to the field name instead. Alternatively, specify offsets explicitly and use field references.

C Language Issues

None.

Language Issues

Generally, debugger support for C++ is limited to programming constructs that are common to C, although support for basic C++ features (such as references, classes and class members) is available.

Specific examples of these and other problems are noted below:

Problem: The debugger shows mangled symbol names, not unmangled ones.

Workaround: Search for the mangled name in the file [.cxx_repository]cxx$demangler_db. This displays a line containing the mangled and unmangled names.

Problem: Examining a data member in an anonymous union results in a debugger access violation.

Workaround: Examine the entire structure.

COBOL Language Issues

Problem: Fixed-point types (type FLOATING with SCALE) are not supported in the Debugger.

Workaround: Examine the data as an integer and manually adjust the value by the appropriate scale.

Problem: Condition variable (level 88) data declarations are not supported.

Workaround: Examine the data as an integer and manually interpret the value according to the condition declaration.

Fortran Language Issues

Problem: Support for LOGICAL*n, REAL*16 and COMPLEX*n data types is incomplete. Examining data in any of these types does not display a value.

Workaround: Use EXAMINE/TYPE=(name) to override the default type.

MACRO-32 Language Issues

MACRO-32 is a compiled language on OpenVMS I64 systems. Because the Itanium® architecture has different hardware register usage than either Alpha or VAX systems, the MACRO-32 compiler must transform source code references to Alpha and VAX registers into a compatible register reference on Itanium®. For example, a reference to R0 in the source program is transformed by the MACRO-32 compiler into Itanium® register R8. The register mapping that is used by MACRO-32 is displayed in Table 3-3:

Table 3-3 Alpha to Itanium® Register Mapping

Alpha Integer Register	Itanium® General Register	Alpha Integer Register	Itanium® General Register
0	8	16	14
1	9	17	15
2	28	18	16
3	3	19	17
4	4	20	18
5	5	21	19
6	6	22	22
7	7	23	23
8	26	24	24
9	27	25	25
10	10	26	No mapping
11	11	27	No mapping
12	30	28	No mapping
13	31	29	29
14	20	30	12
15	21	31	0

HP COBOL T2.8 Release Notes

The following list describes the restrictions and known problems using COBOL T2.8 with OpenVMS I64 systems:

- When installing COBOL on an OpenVMS I64 system, set the default directory to a writable directory, not the DVD media, to allow the IVP to succeed.
- The native compiler and RTL require OpenVMS I64 Version 8.1 or higher.
- The COPY FROM DICTIONARY and DML are not supported.
- Exception handling , including ON SIZE ERROR, is not fully operatonal on OpenVMS I64 systems. For best results, use ON SIZE ERROR on arithmetic statements and ON EXCEPTION on CALL statements. However, ON SIZE ERROR generally is operational just for simple arithmetic expressions. In most cases, arithmetic expressions that raise an exception will not display a run-time diagnostic if you omit ON SIZE ERROR. Also, CALL exception handling is not currently operational unless ON EXCEPTION is used.

- Floating point defaults to /FLOAT_IEEE_FLOAT. Because all floating point operations are done using IEEE floating, results with non-IEEE floating point datatypes are subject to these conversions and precision of IEEE operations. Thus, they are not 100% compatible between OpenVMS I64 and OpenVMS Alpha systems.
- Support for user-written condition handlers is not yet implemented. This means calling LIB$ESTABLISH is not yet supported.
- /DEBUG with COPY is not fully implemented.
- In some cases, compiling /DEBUG results in an expected %SYSTEM-F-INTDIV diagnostic at run time. The workaround is to compile /NODEBUG.
- Compiling /DEBUG suppresses run-time traceback information. The workaround is to compile /NODEBUG.
- /ANALYSIS_DATA support has not yet been tested.
- Tape support has not yet been tested.

See the HP COBOL T2.8 Release Notes for more information.

HP MACRO for OpenVMS Release Notes

The native MACRO compiler has been provided with OpenVMS I64 Evaluation Release Version 8.1. It uses the same exact DCL commands as the MACRO compiler on OpenVMS Alpha.

Most programes that compile on OpenVMS Alpha should recompile on OpenVMS I64 without modification. However, programs that call routines with nonstandard return values or programs that use the JSB instruction call routines written in other languates must add some new directives to the source file.

At this time, the following features are not available:

- The Packed Decimal emulation package provided by the MACRO-32 compiler does not work properly in V8.1. Programs that use VAX packed decimal instructions will compile and link without error, but the run-time behavior is unpredictable. The behavior may include access violations or random results. This package will be ported for the next release.
- The /TIE DCL qualifier is ignored.
- The .BRANCH_LIKELY and .BRANCH_UNLIKELY directives do not take full advantage of the Itanium® architecture.
- The MACRO-32 compiler:
  - — Generates incorrect code for INSV instructions that have literals for both the size and position operands and the position operand is greater than 31. This problem can be avoided by loading one of the two literals into a scratch register and then using that scratch register as an operand to the INSV instruction.
  - — Does not set R25 when calling the SYS$PAL system services that implement the VAX queue instructions. In general, this does not cause a problem; however, there are rare cases where it may cause a system service error. This problem can be avoided by explicitly moving the appropriate literal into R25 with an explicit MOVL instruction.
  - — Sometimes generates suboptimal debug information for entry points that fall into other entry points. It does not occur all the time and is hard to predict.

— Does not conform to the calling standard with respect to protecting against NaT values in integer registers. In general, most programs will not notice. The compiler will be corrected in the future. In the meantime, ensure that incoming registers to MACRO-32 routines do not contain NaT values.

For more information on the new directives and for detail on the MACRO compiler, see the *HP OpenVMS MACRO Compiler Porting and User's Guide*.

Intel® Assembler Release Notes

All modules written in Intel® assembly language must contain appropriate unwind directives, either by automatic generation using the -Xunwind flag or by explicitly placing unwind directives into the source module. Without accurate unwind information, the operating system's condition handling and exception dispatching does not work, and your program fails in unexpected ways. Programs without accurate unwind information are not supported by the debugger. This is a permanent requirement.

Changes to OpenVMS System Dump Analyzer to Support OpenVMS I64

The following changes are for OpenVMS I64 but some also apply to OpenVMS AlphaServer systems.

New Commands

```
SHOW EXCEPTION_FRAME {address|[/SUMMARY][range]}
```

Displays the contents of the exception frame at the given address (which is rounded down to an octaword-aligned address), or displays a one-line summary of all exception frames found on all applicable stacks.

Please note the following:

- The /SUMMARY qualifier is the default.
- SHOW EXCEPTION and SHOW EXCEPTION <range> imply /SUMMARY.
- If a range (either <start:end> or <start;length>) is given, then that range is searched instead of the stacks.
- Under some circumstances, the exception frame of the actual bugcheck is copied (by bugcheck) to the system stack for the CPU. Since this stack is also searched, multiple hits may occur for this exception frame.

Example 3-1 SHOW EXCEPTION

```
SDA> show exception

Exception Frame Summary
-----------------------
```

```
 Exception Frame  Type    Stack   IIP / Ret_Addr   Trap_Type / Code_Address
----------------- ----    -----  ----------------- ------------------------
00000000.7FF43BD0 INTSTK  Kernel 00000000.00020150 00000008  Access control *
00000000.7FF43F40 SSENTRY Kernel 00000000.000200B0 806958B0  PROCESS_MANAGEMENT+658B0
FFFFFFFF.872DFD00 INTSTK  System FFFFFFFF.804D0980 00000041  Bugcheck Breakpoint Trap
```

SHOW SWIS [/RING_BUFFER [/CPU=n]]

The SHOW SWIS command without the /RING_BUFFER qualifier displays the addresses of the SWIS (SoftWare Interrupt Services) data structure for each CPU. SHOW SWIS/RING_BUFFER displays the SWIS ring buffer (also known as the SWIS log), most recent entry first, and assigns meaning to some values (for example, trap type, system service being invoked). For best results, use READ/EXEC or READ/IMAGE SYS$PUBLIC_VECTORS first so that the system service code addresses are recognized. If you specify /CPU=n on the command, only the records for that CPU are displayed.

Example 3-2 SHOW SWIS/RING_BUFFER

[Note: Lines are numbered for the example only, to show their continuation below.]

```
      SDA> read/exec/nolog
      SDA> define ssentry 8692B8F0
      SDA> define intstk 8692B9F0
      SDA> show swis/ring_buffer

      SWIS ring buffer for all CPUs
      -----------------------------

                        8192. entries: Most recent first

[1.]   Clock    Data 1    Data 2        Data 3       CPU  Ident
.     -------- --------  --------  ----------------- --- --------
.     2CEDAD3C 82D66400a 83814080  FFFFFFFF.86B04000  00 SWPCXout
.     2CEDA929 82D66400a 83814080  FFFFF802.0EE370A8  00 SWPCTXin
[5.]  2CED9F16 0000001F  0000001F  FFFFFFFF.8046C270a 00 RaisIPL
.     2CED928F 8692B8F0a 00000000  FFFFFFFF.8046B760b 00 SSSwRet
.     2CED8FED 8692B8E0  00000000  0000002C.DC0351F2  00 RetKSrvc
.     2CED8B2E 8692B8F0a 06900660b FFFFFFFF.8046B760c 00 EntKSrvc
.                                                       EntKSrvc
[10.] 2CED72C1 8692B9F0a 00000000  FFFFFFFF.8692BFC0b 00 ExcpDsp2
.     2CED70B4 8692B9F0a 00000041b FFFFFFFF.80322F50c 00 ExcpDisp
.                                                       ExcpDisp
.     2CED6E84 00000001  00000000  00000000.0001001Fa 00 GetDpth
.     2CED6822 00000016  0000001F  FFFFFFFF.80322EB0a 00 RSetIPL
[15.] 2CED62F0 8692BCF0a 00000003  FFFFFFFF.8066C000b 00 IPDisp
```

```
[Lines 1 to 15 continued.]

[1.]         Symbolized value 'a'             Symbolized value 'b' & 'c'
.     ---------------------------------- ----------------------------------
.     BUG$GQ_HWPCB
.     BUG$GQ_HWPCB
[5.]  EXE$BUGCHECK_SWAPPED_C+000E0
.     SSENTRY                            EXE$BUGCHECK_CONTINUE_C+003C0.
.     SSENTRY                            SYS$RPCC_64_C
.                                        EXE$BUGCHECK_CONTINUE_C+003C0
[10.] INTSTK                             INTSTK+005D0
.     INTSTK                             Bugcheck Breakpoint Trap
.                                        SYSTEM_SYNCHRONIZATION_MIN+42F50
.     LNM$C_DEL_OVERLAY+0001B
.     SYSTEM_SYNCHRONIZATION_MIN+42EB0
[15.] INTSTK+00300                       SCH$IDLE_C+00290
```

SHOW UNWIND [address|/ALL]

By default, SHOW UNWIND displays the master unwind table for system space. If /ALL is given, the details of every system unwind descriptor are displayed. If an address is given, the unwind descriptor for the PC (IIP) is located and displayed. The address can be in system or process space.

See also the new qualifier SHOW PROCESS /UNWIND[=ALL] below

Example 3-3 SHOW UNWIND Sample 1

```
SDA> show unwind
System Unwind Table
-------------------
 Page Header VA          Entries            Region ID
-----------------   -----------------   -----------------
FFFFFFFF.7FFFC000   00000000.00000018   00000000.00000000
FFFFFFFF.7FFFA000   00000000.00000018   00000000.00000000
FFFFFFFF.7FFF8000   00000000.00000018   00000000.00000000
FFFFFFFF.7FF44000   00000000.00000018   00000000.00000000
FFFFFFFF.7F7A0000   00000000.00000018   00000000.00000000
FFFFFFFF.7F56C000   00000000.00000006   00000000.00000000

Image name                    Code Base VA       UT Base VA     Unwind Info Base  Flags
```

```
      MUTE VA         Mode      Code End VA         UT Size             GP
--------------------------- ---------------- ---------------- ----------------- ----------
EXCEPTION_MON               FFFFFFFF.80480000 FFFFFFFF.82D53800 FFFFFFFF.82D53800
 FFFFFFFF.7FFFC020 00000000 FFFFFFFF.8055CDCF 00000000.00002AD8 FFFFFFFF.82F6F400

EXCEPTION_MON               FFFFFFFF.86AB0000 FFFFFFFF.86AB4000 FFFFFFFF.86AB4000 Obsolete
 FFFFFFFF.7FFFC170 00000000 FFFFFFFF.86AB207F 00000000.00000060 FFFFFFFF.82F6F400

IO_ROUTINES_MON             FFFFFFFF.80560000 FFFFFFFF.82D78600 FFFFFFFF.82D78600
 FFFFFFFF.7FFFC2C0 00000000 FFFFFFFF.8064A7AF 00000000.00004B00 FFFFFFFF.82FA2800

IO_ROUTINES_MON             FFFFFFFF.86AB6000 FFFFFFFF.86AB8000 FFFFFFFF.86AB8000 Obsolete
 FFFFFFFF.7FFFC410 00000000 FFFFFFFF.86AB73AF 00000000.000000A8 FFFFFFFF.82FA2800

SYSDEVICE                   FFFFFFFF.80650000 FFFFFFFF.82DA7A00 FFFFFFFF.82DA7A00
 FFFFFFFF.7FFFC560 00000000 FFFFFFFF.8065E90F 00000000.00000240 FFFFFFFF.82FA9400
```

Example 3-4 SHOW UNWIND Sample 2

```
SDA> show unwind 00000000.00020130
Unwind Table Entry for 00000000.00020130
----------------------------------------
Image name: X
MUTE VA:            000007FD.BFFC62C0   Mode:                        00000001
Code Base VA:       00000000.00020000   Code End VA:        00000000.000201FF
UT Base VA:         00000000.00030000   UT Size:            00000000.00000030
Unwind Info Base:   00000000.00030000   GP:                 00000000.00240000
Flags:                            0000
Unwind Descriptor:  00000000.00030090   PC range = 00000000.00020130:00000000.000201DF
    Unwind Descriptor flags:            No handler present, No OSSD present
    Unwind descriptor records:          R1 Region Header: Short Prologue,
                                          PC range = 00000000.00020130:00000000.00020131
                                           P7: MEM_STACK_V PC=00000000.00020131
                                           P3: PSP_GR      R41
                                           P3: PFS_GR      R40
                                        R1 Region Header: Short Body,
                                          PC range = 00000000.00020132:00000000.000201B0
                                           B1: Short Label_State LABEL=00000001
                                           B2: Short Epilogue ECOUNT=00000000
```

```
                                                    PC=00000000.000201A0
                                  R1 Region Header: Short Body,
                                    PC range = 00000000.000201B1:00000000.000201D1
                                     B1: Short Copy_State LABEL=00000001
```

New Qualifiers to Existing Commands

EVALUATE /IFS /PFS /ISR /PSR /FPSR

Example 3-5 EVALUATE/PFS

```
SDA> eval/pfs 00000000.000013AF
        PPL     PEC     RRB.PR   RRB.FR   RRB.GR      SOR        SOL            SOF
        0        0.       0.       0.       0.        0.    39. (32-70)   47. (32-78)
```

Example 3-6 EVALUATE/PSR

```
SDA> eval/psr 00001410.0A026010
        RT  TB  LP  DB  SI  DI  PP  SP  DFH DFL DT  PK  I   IC  MFH MFL AC  BE  UP
        1   0   1   0   0   0   0   0   0   0   1   0   1   1   0   1   0   0   0
        IA  BN  ED  RI  SS  DD  DA  ID  IT  MC  IS  CPL
        0   1   0   2   0   0   0   0   1   0   0   0
```

EXAMINE /IFS /PFS /ISR /PSR /FPSR

Formats the value or location as the specified register.

FPSR: Floating-point Status Register
IFS: Interruption Function State
ISR: Interruption Status Register
PFS: Previous Function State
PSR: Processor Status Register

Example 3-7 EXAMINE/PFS

```
SDA> exam/pfs 7FF43C10
        PPL     PEC     RRB.PR   RRB.FR   RRB.GR      SOR        SOL            SOF
        0        0.       0.       0.       0.        0.    23. (32-54)   31. (32-62)
```

Example 3-8 EXAMINE/PSR

```
SDA> exam/psr 7FF43C78
        RT  TB  LP  DB  SI  DI  PP  SP  DFH DFL DT  PK  I   IC  MFH MFL AC  BE  UP
        1   0   1   0   0   0   0   0   1   0   1   0   1   1   0   1   0   0   0
        IA  BN  ED  RI  SS  DD  DA  ID  IT  MC  IS  CPL
        0   1   0   1   0   0   0   0   1   0   0   0
```

FORMAT /NOSYMBOLIZE

If /NOSYMBOLIZE is specified, no attempt will be made to symbolize the contents of any field in a structure. This has proved useful if the loaded execlet or activated image lists are corrupted, since symbolization relies on these lists.

SHOW CALL

The address is the invocation context handle of the frame. For OpenVMS I64, the default is the handle of the current procedure.

SHOW CALL/ALL

Displays all call frames beginning at the current frame and continues until reaching bottom-of-stack (equivalent to SHOW CALL with repeated executions of SHOW CALL/NEXT). SHOW CALL/NEXT_FRAME replaces SHOW CALL/NEXT_FP.

SHOW CALL /SUMMARY

Provides a one-line summary for each call frame including exception frames, system-service entry frames, ASTs, KPBs, and so on, until reaching the bottom stack.

Example 3-9 SHOW CALL/SUMMARY

```
SDA> sh call/summary

Call Frame Summary

------------------

     Frame Type              Handle             Current PC

--------------------    -----------------    -----------------

Exception Dispatcher    00000000.7FF43EB0    FFFFFFFF.8049E160  EXCEPTION_MON+5E360

Register Stack Frame    00000000.7FF12180    00000000.000122C0  KP_SAMPLE+122C0

Memory Stack Frame      00000000.7FF43ED0    FFFFFFFF.8066B440  EXE$CMKRNL_C+00330

Memory Stack Frame      00000000.7FF43F20    FFFFFFFF.80194890  EXE$SS_DISP_C+00400

SS Dispatcher           00000000.3FFFDFC0    FFFFFFFF.8018D240  SWIS$ENTER_KERNEL_SERV*

Register Stack Frame    000007FD.BFF58000    00000000.000124C0  KP_SAMPLE+124C0

KP Start Frame          00000000.7AC95A20    FFFFFFFF.80161670  EXE$KP_START_C+003C0

Memory Stack Frame      00000000.7AC95B50    00000000.00012CE0  KP_SAMPLE+12CE0

Memory Stack Frame      00000000.7AC95BC0    00000000.000126F0  KP_SAMPLE+126F0

Base Frame              00000000.7AC95BE0    00000000.7ADE0BB0  DCL+82BB0

Bottom of stack
```

SHOW PARAMETER /OBSOLETE

Displays all the obsolete system parameters. (As with the SYSGEN & SYSMAN changes, SDA will only display obsolete parameters if they are named explicitly without wildcards, or if /OBSOLETE is specified).

SHOW PROCESS /UNWIND[=ALL]

SHOW PROCESS /UNWIND will display the master unwind table for the process. SHOW PROCESS /UNWIND=ALL will display the details of every process unwind descriptor. If you need to look at the unwind data for a specific PC in process space, use SHOW UNWIND <address>.

SHOW PROCESS/IMAGE=<name>

Will display the details of the sections for just the specified image. SHOW PROCESS/IMAGE continues to provide a one-liner summary of all the images; SHOW PROCESS/IMAGE=ALL continues to provide details for all images.

Changes to Existing Commands

EVALUATE/PTE

Page protections changed.

EXAMINE/PTE

Page protections changed.

EXAMINE/INSTRUCTION

Output has changed. SDA always displays entire bundles of instructions, not individual slots.

FORMAT

The FORMAT command now traverses structures by quadwords instead of longwords (shorter fields continue to be displayed separately). Also, any sequence of unlabeled quadwords containing the same value as the previous quadword is replaced by ellipses.

Example 3-10 FORMAT

```
SDA> read sysdef
SDA> read/exec
SDA> format 82D28900
FFFFFFFF.82D28900   SPL$L_OWN_CPU                          00000000
FFFFFFFF.82D28904   SPL$L_OWN_CNT                 FFFFFFFF
FFFFFFFF.82D28908   SPL$W_SIZE                           0100
FFFFFFFF.82D2890A   SPL$B_TYPE                        4F
FFFFFFFF.82D2890B   SPL$B_SUBTYPE                     01
FFFFFFFF.82D2890C   SPL$L_SPINLOCK              00000000
FFFFFFFF.82D28910   SPL$L_RANK                          00000000
FFFFFFFF.82D28914   SPL$B_IPL                     1F
                    SPL$L_IPL
FFFFFFFF.82D28915                              000000
FFFFFFFF.82D28918   SPL$L_RLS_PC                        00000000
FFFFFFFF.82D2891C   SPL$L_BUSY_WAITS           00000000
FFFFFFFF.82D28920   SPL$L_WAIT_CPUS                     00000000
```

```
FFFFFFFF.82D28924   SPL$L_WAIT_PC                   00000000
FFFFFFFF.82D28928   SPL$Q_SPINS                     00000000.00000000
FFFFFFFF.82D28930   SPL$Q_ACQ_COUNT                 00000000.00000396
FFFFFFFF.82D28938   SPL$L_TIMO_INT                          000186A0
FFFFFFFF.82D2893C   SPL$PS_SHARE_ARRAY              00000000
FFFFFFFF.82D28940   SPL$PS_SHARE_LINK               00000000.00000000
                    SPL$T_NAME
FFFFFFFF.82D28948                                   00000000.00000000
FFFFFFFF.82D28950   SPL$Q_RELEASE_COUNT             00000000.00000396
FFFFFFFF.82D28958   SPL$Q_HISTORY_BITMASK           00000000.00000000
FFFFFFFF.82D28960   SPL$Q_ABUSE_THRESHOLD           00000000.00000000
FFFFFFFF.82D28968   SPL$Q_FLAGS                     00000000.00000000
FFFFFFFF.82D28970                                   00000000.00000000
       ...                                                 ...
FFFFFFFF.82D28980   SPL$Q_ABUSE_BITMASK            00000000.00000000
FFFFFFFF.82D28988                                   00000000.00000000
       ...                                                 ...
FFFFFFFF.82D289B8                                            00000000
FFFFFFFF.82D289BC   SPL$L_VEC_INX                   0000000C
FFFFFFFF.82D289C0   SPL$L_OWN_PC_VEC                        8013DCD0    ERL$WAKE_C+00370
FFFFFFFF.82D289C4                                   8013E480            ERL$WAKE_C+00B20
FFFFFFFF.82D289C8                                   8013DDF0.8013DCD0
FFFFFFFF.82D289D0                                   00022A40.00022480
FFFFFFFF.82D289D8                                   00023450.00023130
FFFFFFFF.82D289E0                                   00022A40.00022480
FFFFFFFF.82D289E8                                   00023450.00023130
FFFFFFFF.82D289F0                                   8013E480.8013DCD0
       ...                                                 ...

                    SPL$C_LENGTH
```

READ

Accepts quoted filenames for access to images on ODS-5 disks with lowercase or compound characters in their names. /NOLOG is now the default for the READ command (Previously, the default was /LOG.)

SET CPU n

Under ANALYZE/SYSTEM, SET CPU now lists the database address for the specified CPU before exiting with the "command not valid on the running system" error message.

SHOW CALL_FRAME

Display is new.

Walking the full call chain may not be possible because some OpenVMS systems have no unwind information; the SDA error message indicates this clearly.

On I64, there is a change in how the registers display with the SHOW CALL command. On OpenVMS Alpha, the command displays the register value that was saved when executing the call, which is the saved value from the previous procedure. On OpenVMS I64, it is the value in the current procedure.

SHOW CPU [n]

Previously disallowed under ANALYZE/SYSTEM, SHOW CPU now lists the database address(es) for the specified CPU or all CPUs.

SHOW CRASH

New exception display.

If a KRNLSTAKNV or DEBUGCRASH bugcheck has occurred, the exception frame from the original exception is also displayed. SHOW CRASH displays the original exception in process dumps and in system dumps; it now additionally displays all CPU database addresses.

Example 3-11 SHOW CRASH

```
SDA> show crash

System crash information

------------------------

Time of system crash:  1-DEC-2003 13:31:10.50

Version of system: OpenVMS I64 Operating System, Version XA2T-J2S

System Version Major ID/Minor ID: 3/0

System type: HP rx2600  (900MHz/1.5MB)


Crash CPU ID/Primary CPU ID:  01/00

Bitmask of CPUs active/available:  00000003/00000003

CPU bugcheck codes:

        CPU 01 -- database address 8396DD80 -- SSRVEXCEPT, Unexpected system service exception

        1 other -- CPUEXIT, Shutdown requested by another CPU

                CPU 00 -- database address 83864000


System State at Time of Original Exception
------------------------------------------

Exception Frame at 00000000.7FF43BD0
------------------------------------

    IPL             =                 00

    TRAP_TYPE       =           00000008      Access control violation fault

    IVT_OFFSET      =           00000800      Data TLB Fault

    IIP             =  00000000.00020120      SYS$K_VERSION_08+00100

    IIPA            =  00000000.00020110      SYS$K_VERSION_08+000F0
```

```
    IFA             =  00000000.00000000

[next 4 lines: spaces compressed for example]
IPSR = 00001010.0A0A6010 RT TB LP DB SI DI PP SP DFH DFL DT PK I IC MFH MFL AC BE UP
                         1  0  1  0  0  0  0  0  1   0   1  0  1 1  0   1   0  0  0
                         IA BN ED RI SS DD DA ID IT  MC  IS CPL
                         0  1  0  0  0  0  0  0  1   0   0  0

    PREVSTACK       =                00
    BSP             =  00000000.7FF12240
    BSPSTORE        =  00000000.7FF120C0
    BSPBASE         =  00000000.7FF120C0
    RNAT            =  00000000.00000000
    RSC             =  00000000.00000003      LOADRS   BE   PL   MODE
                                              0000     0    0    Eager

[next 2 lines: spaces compressed for example]
PFS = 00000000.00000B9F PPL PEC RRB.PR RRB.FR RRB.GR SOR     SOL         SOF
                        0    0.    0.     0.     0.   0.  23. (32-54) 31. (32-62)

    FLAGS           =                00
    STKALIGN        =           000002D0
    PREDS           =  00000000.FF562AA3
    IHA             =  FFFFFFFF.7FF3E120
    INTERRUPT_DEPTH =                00

    ISR             =  00000804.00000000      ED  EI  SO  NI  IR  RS  SP  NA  R   W   X   CODE
                                              1   0   0   0   0   0   0   0   1   0   0   0000

    ITIR            =  00000000.FFFF0934      KEY       PS
                                              FFFF09    0D

[next 2 lines: spaces compressed for example]
IFS = 80000000.00000593 Valid RRB.PR RRB.FR RRB.GR SOR    SOL        SOF
                        1        0.     0.     0.   0. 11. (32-42)19. (32-50)

    B0              =  FFFFFFFF.80241AE0      AMAC$EMUL_CALL_NATIVE_C+00340
    B1              =  80000000.FFD643B0
```

```
B2              =  00000000.00000000
B3              =  00000000.00000000
B4              =  00000000.00000000
B5              =  00000000.7FF43E38
B6              =  00000000.00020110      SYS$K_VERSION_08+000F0
B7              =  FFFFFFFF.80A28170      NSA$CHECK_PRIVILEGE_C

GP              =  00000000.00240000
R2              =  FFFFFFFF.839B8098      PSB+00058
R3              =  E0000000.00000068
R4              =  FFFFFFFF.839731C0      PCB
R5              =  00000000.00000008
R6              =  00000000.7FF43F40
R7              =  00000000.00000002
R8              =  00000000.00010000      SYS$K_VERSION_07
R9              =  00000000.00000020
R10             =  00000000.0000003E
R11             =  00000000.00000001

KSP             =  00000000.7FF43EA0

R13             =  00000000.00000000
R14             =  00000000.00040008      UCB$M_SUPMVMSG+00008
R15             =  00000000.00020110      SYS$K_VERSION_08+000F0
R16             =  FFFFFFFF.802417A0      AMAC$EMUL_CALL_NATIVE_C
R17             =  00000000.00010004      UCB$M_DELETEUCB+00004
R18             =  00000000.00040000      UCB$M_CHAN_TEAR_DOWN
R19             =  00000000.00040000      UCB$M_CHAN_TEAR_DOWN
R20             =  00000000.7FF43F38
R21             =  00000000.7FF43F80
R22             =  00000000.00040000      UCB$M_CHAN_TEAR_DOWN
R23             =  00000000.00000000
R24             =  00000000.00000000
R25             =  00000000.00000000
R26             =  00000000.00000000
R27             =  00000000.FF565663
R28             =  00000000.00000003
R29             =  00000000.7FF43EA0
```

```
R30             =  000007FD.C0000300
R31             =  FFFFFFFF.806549D0      PROCESS_MANAGEMENT_MON+677D0

R32             =  00000000.7AC9DBC0
R33             =  00000000.00000001
R34             =  00000000.7FFCF88C      MMG$IMGHDRBUF+0008C
R35             =  FFFFFFFF.83973528      ARB+00230
R36             =  00000000.00000000
R37             =  00000000.00000000
R38             =  FFFFFFFF.80A28410      NSA$CHECK_PRIVILEGE_C+002A0
R39             =  00000000.00000915
R40             =  FFFFFFFF.82D01640      SYSTEM_PRIMITIVES+00221440
R41             =  00000000.00000B9F
R42             =  00000000.7FF43EA0

R43/OUT0        =  00000000.7FFCF87C      MMG$IMGHDRBUF+0007C
R44/OUT1        =  E0000000.00000068
R45/OUT2        =  00000000.00000000
R46/OUT3        =  00000000.FF561663
R47/OUT4        =  00000000.7FFCDA68      CTL$AG_CLIDATA
R48/OUT5        =  00000000.7FFCDBE8      CTL$AG_CLIDATA+00180
R49/OUT6        =  00000000.00000003
R50/OUT7        =  FFFFFFFF.839731C0      PCB

NATMASK         =              003A
NATS            =  00000000.00000000
CSD             =  CFFFFFFF.00000000
SSD             =  CCCC0BAD.BAD0CCCC
LC              =  00000000.00000000
EC              =  00000000.00000000

FPSR            =  0009804C.0270033F      SF3    SF2    SF1    SF0    TRAPS
                                          004C   004C   004E   000C   3F

F6              =  0FFC9.C0000000.00000000
F7              =  1003E.00000000.00000018
F8              =  1000B.FF000000.00000000
F9              =  10007.A8000000.00000000
```

```
    F10             =  10003.C2492492.49249249
    F11             =  0FFF6.C30C30C3.0C30C30C

    PPREVMODE       =               03

Instruction Stream:
-------------------
                                  { .mfb
         SYS$K_VERSION_08+000E0:                nop.m       000000
                                                nop.f       000000
                                                br.ret.sptk.many b0 ;;
                                  }
                                  { .mii
         SYS$K_VERSION_08+000F0:                alloc       r41 = ar.pfs, 0B, 08, 00
                                                mov         r29 = r12
                                                mov         r42 = r12
                                  }
                                  { .mmi
   PC => SYS$K_VERSION_08+00100:                ld4         r24 = [r0] ;;
                                                nop.m       000000
                                                sxt4        r24 = r24 ;;
                                  }
                                  { .mii
         SYS$K_VERSION_08+00110:                nop.m       000000
                                                sxt4        r14 = r24 ;;
                                                cmp.eq      p6, p7 = r14, r0
                                  }
                                  { .mfb
         SYS$K_VERSION_08+00120:                nop.m       000000
                                                nop.f       000000
                                           (p6) br.cond.dpnt.few 0000060
                                  }

Signal Array
------------
Length = 00000005
Type   = 0000000C
Arg    = 00000000.00000000
```

```
Arg    = 00000000.00000000
Arg    = 00000000.00020120
Arg    = 00000000.00000003
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=0000000000000000,
                                                    PC=0000000000020120, PS=00000003

CPU 01 Processor state at time of SSRVEXCEPT bugcheck
-----------------------------------------------------
CPU 01 reason for Bugcheck: SSRVEXCEPT, Unexpected system service exception
Process currently executing on this CPU: SYSTEM
Current image file: IPFEX3$DKB200:[SYS0.][SYSMGR]X.EXE;2
Current IPL: 0  (decimal)

CPU database address: 8396DD80
CPUs Capabilities:    QUORUM,RUN
Exception Frame at 00000000.7FF435B0
-------------------------------------
    IPL             =               00
    TRAP_TYPE       =         00000041     Bugcheck Breakpoint Trap
    IVT_OFFSET       =           00002C00      Break Instruction
     IIP             =  FFFFFFFF.80491E90      EXCEPTION_MON+5E690
     IIPA            =  FFFFFFFF.80491E80      EXCEPTION_MON+5E680
     IFA             =  00000000.00030000      SYS$K_VERSION_01
     IIM             =  00000000.00100002      BREAK$C_SYS_BUGCHECK
     PPREVMODE       =                 00
    KR0             =  00000000.00000000
     KR1             =  00000000.00000000
     KR2             =  00000000.00000000
     KR3             =  00000000.00000003
     KR4             =  00000000.00000000
     KR5 (Next Timer) =  000000BC.DEA95C24
     KR6 (CPUdb VA)   =  FFFFFFFF.8396DD80
     KR7 (Slot VA)    =  FFFFFFFF.86910000
     KSP             =  00000000.7FF43880
     ESP             =  00000000.7FF68000
     SSP             =  00000000.7FFAC000
     USP             =  00000000.7AC9DB60
```

```
                No spinlocks currently owned by CPU 01

CPU 00 Processor state at time of CPUEXIT bugcheck
--------------------------------------------------
CPU 00 reason for Bugcheck: CPUEXIT, Shutdown requested by another CPU
Process currently executing on this CPU:   None
Current IPL: 31  (decimal)
CPU database address: 83864000
CPUs Capabilities:    PRIMARY,QUORUM,RUN

Exception Frame at FFFFFFFF.8696F9F0
------------------------------------
    IPL              =                1F
    TRAP_TYPE        =          00000041      Bugcheck Breakpoint Trap
    IVT_OFFSET       =          00002C00      Break Instruction
    IIP              =  FFFFFFFF.802F62F0      SYSTEM_SYNCHRONIZATION+43BF0
    IIPA             =  FFFFFFFF.802F62F0      SYSTEM_SYNCHRONIZATION+43BF0
    IFA              =  FFFFFFFF.86A280C0

    IIM              =  00000000.00100002      BREAK$C_SYS_BUGCHECK

    PPREVMODE        =                00

    KR0              =  00000000.203D0000
    KR1              =  00000000.60000000
    KR2              =  00000000.00000000
    KR3              =  00000000.0001001F
    KR4              =  00000000.00000000
    KR5 (Next Timer) =  000000C4.FDFE03C8
    KR6 (CPUdb VA)   =  FFFFFFFF.83864000
    KR7 (Slot VA)    =  FFFFFFFF.8690F000

    KSP              =  FFFFFFFF.8696FCC0
    ESP              =  FFFFFFFF.86971000
    SSP              =  FFFFFFFF.86957000
    USP              =  FFFFFFFF.86957000

                No spinlocks currently owned by CPU 00
```

SHOW DEVICE

Accepts an optional colon on the end of a given device name. All qualifiers specific to Memory Channel are disabled for I64.

SHOW EXECUTIVE

GP (global pointer) added.

Example 3-12 SHOW EXECUTIVE

```
SDA> show executive


VMS Executive layout summary
----------------------------
Image                                  LDRIMG   SeqNum         GP           SymVec
-------------------------------------- -------- --------- ----------------- --------
SYS$RTTDRIVER                         83A5DA40 0000007E FFFFFFFF.832C2600
SYS$CTDRIVER                          83A57A40 0000007C FFFFFFFF.832BEA00
NDDRIVER                              83A198C0 0000007A FFFFFFFF.832B6E00
...
SYS$PLATFORM_SUPPORT                  838166C0 00000004 FFFFFFFF.82E4A400
SYS$BASE_IMAGE                        838164C0 00000002 FFFFFFFF.82E16600 82C16600
SYS$PUBLIC_VECTORS                    838162C0 00000000 FFFFFFFF.82E00400 82C00400
```

SHOW EXECUTIVE/ALL

Displays data for all loadable images. This is the default.

Example 3-13 SHOW EXECUTIVE/ALL

```
[Note: Lines are numbered for the example only, to show their continuation below.]

      SDA> show executive/all

      VMS Executive layout
      --------------------
[1.]  Image                                    Base              End
.     -------------------------------------- ----------------- -----------------
.     SYS$RTTDRIVER
.         Data (read only)                  FFFFFFFF.830BF800 FFFFFFFF.830BF89F
[5.]      Code                              FFFFFFFF.8139D200 FFFFFFFF.813A7BDF
.         Data (read only)                  FFFFFFFF.830BFA00 FFFFFFFF.830C2167
.         Data (read/write)                 FFFFFFFF.830C2200 FFFFFFFF.830C254F
.         Short data                        FFFFFFFF.830C2600 FFFFFFFF.830C2B3F
.             Linked 10-NOV-2003 11:38          LDRIMG 83A5DA40   SeqNum 0000007E
```

```
...

[10.] SYS$PUBLIC_VECTORS
.         Code                                FFFFFFFF.80000000 FFFFFFFF.8000006F
.         Data (read only)                    FFFFFFFF.82C00000 FFFFFFFF.82C000AF
.         Data (read/write)                   FFFFFFFF.82C00200 FFFFFFFF.82C00217
.         Short data                          FFFFFFFF.82C00400 FFFFFFFF.82C0851F
[15.]         Linked 10-NOV-2003 11:36          LDRIMG 838162C0   SeqNum 00000000

[Lines 1 to 15 continued.]

[1.]       Length       ImageOff  SymVec
    -----------------   --------  -------.
.     00000000.000000A0 00010000
[5.]  00000000.0000A9E0 00014000
.     00000000.00002768 00020000
.     00000000.00000350 00024000
.     00000000.00000540 00030000
.           GP FFFFFFFF.832C2600
...
[10.]                           82C00400
.     00000000.00000070 00010000
.     00000000.000000B0 00014000
.     00000000.00000018 00018000
.     00000000.00008120 0001C000
[15.]       GP FFFFFFFF.82E00400
```

SHOW PAGE

SHOW PAGE with an address or range of addresses now accepts addresses in process space. Previously, you had to use SHOW PROCESS/PAGE <address> (which continues to work).

Page protections changed.

SHOW PROCESS/IMAGE

GP (global pointer) added.

SHOW PROCESS/REGISTERS

For an ANALYZE/SYSTEM command, output from the SHOW PROCESS/REGISTERS command is disabled.

SHOW PROCESS/PAGE

Page protections changed.

CLUE Commands not Ported

The following CLUE commands have not yet been ported to OpenVMS I64 and will return a message to that effect:

- CLUE CALL
- CLUE REGISTER
- CLUE FRU
- CLUE ERRLOG

DELTA Release Notes

The DELTA debugger has not yet been ported to the OpenVMS I64 operating system.

XDELTA Release Notes

In general, the XDelta Debugger for OpenVMS I64 systems behaves the same as on OpenVMS Alpha systems. This release note assumes that you are already familiar with XDELTA on OpenVMS Alpha.

New Features

XDELTA on OpenVMS I64 systems contains the following new features:

- Access to the following Itanium® registers:
  - General registers: R0 through R127
  - Floating registers: FP0 through FP127
  - Application registers: AR0 through AR127
  - Branch registers: BR0 through BR7
  - Control registers: CR0 through CR63
  - Predicate registers: P0 through P63
  - Miscellaneous registers: PC, PS, CFM
  - Software implementation of Alpha hardware registers
- New commands. The ;D and ;T commands are new.
- Better symbolization

General Registers R0 through R127

All general registers are accessible using the name 'Rn', where n is a number between 0 and 127, inclusive. However, the Itanium® architecture limits access to the stacked registers (R32 through R127) based on the size of the current stack frame field SOF in register CFM. For example, if SOF equals 5, then the general

registers that can be accessed are from R0 through R36, inclusive (static registers R0 through R31 and stacked registers R32 through R36). Attempts to access the general registers beyond the size of the stack frame results in an error.

The Itanium® architecture divides the stacked registers into three regions: input, local and output. The size of each region is defined by the `br.call` and `alloc` instructions. XDELTA provides names of the form OUTx as alias names for the stacked output registers, where "x" is a number between 0 and the size of the stacked output region (defined by the SOF and SOL fields in the CFM register). These alias names must be specified in XDELTA using the "P(OUTx)" syntax. The alias names are useful because the mapping from the name OUT0 to the general register Rx varies depending on the number of input and local registers. For example, if there are 3 input and 2 local stacked registers, then OUT0 maps to stacked register R37. XDELTA does not provide alias names for input and local registers.

If a general register's NaT bit is set, XDelta displays the value of the register as "NaT". You can clear the NaT bit by depositing a value to the corresponding general register. NaT bits cannot be set.

Deposits to R0 and the stack pointer (R12) are not allowed.

Floating Registers FP0 through FP127

All floating-point registers are accessible using the name 'FPn', where n is a number between 0 and 127, inclusive.

Floating-point registers are always displayed as 128-bit quantities in Itanium register format.

Deposits to floating-point registers must be performed in multiple steps, because XDELTA's maximum size of a value is a quadword. For example, if the data type setting is quadword (the default), then two deposits are needed: one to FPx and another to FPx+8

Deposits to FP0 and FP1 are not allowed.

Application Registers AR0 through AR127

All 128 application registers are defined by XDELTA, but note that:

- Ignored registers are read as zero and cannot be written
- Reserved registers cannot be read and cannot be written

Application registers are accessed using the name 'ARn', where n is a number between 0 and 127, inclusive (subject to the limits on ignored and reserved registers noted above). Application registers can also be accessed using two additional "alias" names, as defined by the Itanium® architecture. Alias names must be specified using the processor register syntax "P(reg)", where "reg" is the alias name. For example, the following are all names for the backing store pointer register:

- AR17
- P(AR.BSP)
- P(BSP)

Branch Registers BR0 through BR7

All branch registers are available.

Control Registers CR0 through CR63

All 128 control registers are defined by XDelta, but note that:

- Ignored registers are read as zero and cannot be written

- Reserved registers cannot be read and cannot be written

Control registers are accessed using the name 'CRn' where n is a number between 0 and 127 inclusive (subject to the limits noted above). Control registers can also be accessed using two additional "alias" names, as defined by the Itanium® architecture. Alias names must be specified using the processor register syntax "P(reg)", where "reg" is the alias name. For example, the following are all names for the interrupt function state register:

- CR23
- P(CR.IFS)
- P(IFS)

Predicate Registers P0 through P63

The Predicate registers are single-bit registers. The register name "PR" is also defined; it provides access to all predicate registers as a single quadword bitvector.

Miscellaneous Registers: PC, PS, CFM

The PC register is a software-only value composed of the hardware IP (containing the instruction bundle address) and the slot offset (slot number of the instruction within the bundle).

The PS register is an alias name for the IPSR control register (CR16). (This is provided for compatibility with XDELTA on OpenVMS Alpha systems.)

The CFM register contains the current frame marker value.

Alpha Software Registers

XDELTA on OpenVMS I64 provides software-only emulation of certain Alpha hardware registers. These registers are available in XDELTA on I64 systems, just as they are on Alpha, by using the "P(reg)" syntax, where "reg" is one of the following:

ASTEN	ASTSR	DATFX
ESP	FEN	IPIR
IPL	MCES	PCBB
PRBR	RSCC	SIRR
SISR	SSP	TBCHK
TBIA	TBIAP	TBIS
TBISD	TBISI	USP
WHAMI		

Note that the read/write characteristics of these software "registers" on I64 systems are the same as on Alpha systems.

New Commands

The following commands are now available on I64 systems and Alpha systems.

;D Command The ;D command dumps a region of memory. The display is in a format similar to the DCL DUMP command. The command syntax is:

<addr>, <length> ;D

where:

<addr> is the starting address of the dump

<length> is the length in bytes to dump

Both arguments are expressed as hex numbers.

;T Command The ;T command displays the contents of an interrupt stack frame. The command syntax is:

<addr> ;T

where:

<addr> is the address of the interrupt stack frame.

<addr> is an optional argument. If not specified, the ;T command without any argument displays the interrupt stack frame with which XDELTA was invoked.

Better Symbolization

The XDELTA X-registers are used by programmers to hold frequently-used values, like base addresses of images and modules. When displaying breakpoints and other address values, XDELTA prints these values relative to the nearest X-register value. Previously, only certain values were checked for nearness to X-register values.

For example, if X0 contains the value 20000 and a breakpoint is set at address 20100, the ;B command that lists breakpoints shows breakpoint 1 at X0+00000100.

Restrictions

The following Itanium® hardware registers are not supported by XDELTA:

- CPUID
- Debug Data Break Registers
- Debug Instruction Break Registers
- Region Registers
- Protection Key Registers
- Instruction Translation Registers
- Data Translation Registers
- Device Interrupt Control Registers

Differences between XDELTA on I64 systems compared to XDELTA on Alpha systems:

This section lists differences in XDELTA behavior on OpenVMS I64 systems and on OpenVMS Alpha systems.

Interrupting a Running System

To interrupt a running system on OpenVMS I64, press Ctrl/P at the system console. Note that XDELTA must have been previously loaded as described in the section entitled, “Booting OpenVMS from the MP Console” on page 24.When you press Ctrl/P, the system is halted at the current PC and at the current IPL. There is no delay in waiting for the IPL to drop below 14 as on OpenVMS Alpha systems.

Explicitly Linking LIB$SIGNAL from SYS$LIBRARY:STARLET.OLB

If a program references LIB$SIGNAL, at least one other LIB$ routine, and explicitly links against SYS$LIBRARY:STARLET.OLB on the linker command line, you will receive a linker warning. For example:

```
$ TYPE T.MAR
a:: .call_entry
     pushl #1
     calls #1, lib$signal
     calls #0, lib$day
     ret
     .end a
$
$ MACRO T
$
$ LINK T   ! produces no linker warning
$
$ LINK T,SYS$LIBRARY:STARLET.OLB/LIB ! produces a linker warning
%ILINK-W-MULPSC, conflicting attributes for psect _LIB$CODE
              in module LIB$SIGNAL file SYS$COMMON:[SYSLIB]STARLET.OLB;1
```

This error occurs because _LIB$CODE psect declaration inside of LIB$SIGNAL does not specify the SHR psect attribute. This error can safely be ignored or the reference to STARLET.OLB can be removed from the LINK DCL command. The source for LIB$SIGNAL will be fixed in a future release to specify the SHR psect attribute.

If the LINK command references STARLET.OLB because the eventual image will be a protected shared image that is not allowed to make outbound calls, please submit this information to your support channels described in your SDK letter for this kit. Feedback in this area could result in changes to reduce restrictions that are placed upon protected shared images.

A Installation Script

This appendix shows a sample installation script of OpenVMS I64 Evaluation Release Version 8.1. Note, although the script discusses an upgrade, the OpenVMS I64 Evaluation Release Version 8.1 supports only a fresh installation. Also, the installation is from a DVD, not a CD-ROM.

```
    HP OpenVMS Industry Standard 64 Evaluation Release V8.1
    (c) Copyright 1976-2003 Hewlett-Packard Development Company, L.P.


%SMP-I-CPUTRN, CPU #01 has joined the active set.

    Installing required known files...

    Configuring devices...
%EIA0, Auto-negotiation mode assumed set by console
%EIA0, Auto-negotiation started, advertising 100BaseTX Full Duplex
%EIA0, Link partner not auto-negotiation capable, using parallel detection
%EIA0, Half Duplex 100BaseTX connection selected
%EWA0, Auto-negotiation mode assumed set by console
%EWA0, 5701 LOM located in 64-bit, 66-mhz PCI slot
%EWA0, Device type is BCM5703C (UTP) Rev A5 (01050000)
%PKA0, Copyright (c) 2001 LSI Logic, PKM X1.1.01
%PKA0, SCSI Chip is LSI53C1030, Operating mode is LVD
%PKB0, Copyright (c) 2001 LSI Logic, PKM X1.1.01
%PKB0, SCSI Chip is LSI53C1030, Operating mode is LVD

    ************************************************************
    You can install or upgrade the OpenVMS I64 operating system
    or you can install or upgrade layered products that are included
    on the OpenVMS I64 operating system CD-ROM.

    You can also execute DCL commands and procedures to perform
    "standalone" tasks, such as backing up the system disk.

     Please choose one of the following:

        1)  Upgrade, install or reconfigure OpenVMS I64 Version V8.1
        2)  Display products and patches that this procedure can install
        3)  Install or upgrade layered products and patches
        4)  Show installed products
        5)  Reconfigure installed products
        6)  Remove installed products
        7)  Execute DCL commands and procedures
        8)  Shut down this system


Enter CHOICE or ? for help: (1/2/3/4/5/6/7/8/?) 1
    ********************************************************
```

```
    The installation procedure will ask a series of questions.

          () - encloses acceptable answers
          [] - encloses default answers

    Type your response and press the <Return> key.  Type:

          ? - to repeat an explanation
          ^ - to change prior input (not always possible)
          Ctrl/Y - to exit the installation procedure

    There are two choices for installation/upgrade:

    Initialize - Removes all software and data files that were
        previously on the target disk and installs OpenVMS I64.

    Preserve -- Installs or Upgrades OpenVMS I64 on the target disk
        and retains all other contents of the target disk.

   * Note: You cannot use preserve to install OpenVMS I64 on a disk on
        which OpenVMS VAX or any other operating system is installed.

Do you want to INITIALIZE or to PRESERVE? [PRESERVE] initialize

    You must enter the device name for the target disk on which
    OpenVMS I64 will be installed.

Enter device name for target disk: (? for choices) ?

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DAD0:                   Online               0
DKA0:                   Online               0
DKA100:                 Online               0
DKB200:                 Online               0
DQA0:                   Mounted wrtlck       0  I64081          95664    36   1
DQA1:                   Online               1

Enter device name for target disk: (? for choices) dka100
```

```
    DKA100: is now labeled FBR6_XA2T2.

Do you want to keep this label? (Yes/No) [Yes] n

Enter volume label for target system disk: [I64SYS]

    The target system disk can be initialized with On-Disk Structure
    Level 2 (ODS-2) or Level 5 (ODS-5).  (? for more information)

Do you want to initialize with ODS-2 or ODS-5? (2/5/?) 2

    You have chosen to install OpenVMS I64 on a new disk.

    The target system disk, DKA100:, will be initialized
    with structure level 2 (ODS-2).
    It will be labeled I64SYS.
    Any data currently on the target system disk will be lost.

Is this OK? (Yes/No) y

    Initializing and mounting target....

   Creating page and swap files....

    You must enter a password for the SYSTEM account.

   The password must be a minimum of 8 characters in length, and
    may not exceed 31 characters.  It will be checked and verified.
    The system will not accept passwords that can be guessed easily.

    The password will not be displayed as you enter it.

Password for SYSTEM account:

Re-enter SYSTEM password for verification:

Will this system be a member of an OpenVMS Cluster? (Yes/No) y

    When your new system is first booted you will be required to answer
    additional questions in order to configure the OpenVMS Cluster correctly.
```

```
Will this system be an instance in an OpenVMS Galaxy? (Yes/No) n

    For your system to operate properly, you must set two parameters:
    SCSNODE and SCSSYSTEMID.

    SCSNODE can be from 1 to 6 letters or numbers.  It must contain at
    least one letter.

    If you plan to use DECnet, SCSNODE must be the DECnet Phase IV
    node name, or the DECnet-Plus (Phase V) node synonym.

    If you have multiple OpenVMS systems, the SCSNODE on each system
    must be unique.

Enter SCSNODE: nodename

    If you plan to use DECnet, SCSSYSTEMID must be set based on the
    DECnet Phase IV address.

Do you plan to use DECnet (Yes/No) [Yes]:

    DECnet Phase IV addresses are in the format

        DECnet_area_number.DECnet_node_number

    DECnet_area_number is a number between 1 and 63.
    DECnet_node_number is a number 1 and 1023.

    If you plan to use DECnet WITHOUT Phase IV compatible addresses,
    enter 0.0.

Enter DECnet (Phase IV) Address [1.1]: nn.nnn

      SCSSYSTEMID will be set to 20106.

      This was calculated as follows:

          (DECnet_area_number * 1024) + DECnet_node_number
```

```
  Configuring the Local Time Zone

TIME ZONE SPECIFICATION -- MAIN Time Zone Menu            "*" indicates a menu

  0* GMT
  1* AFRICA          12) EET            23) JAPAN          34) ROK
  2* AMERICA         13) EGYPT          24) LIBYA          35) SINGAPORE
  3* ANTARCTICA      14) FACTORY        25) MET            36* SYSTEMV
  4* ASIA            15) GB-EIRE        26* MEXICO         37) TURKEY
  5* ATLANTIC        16) GREENWICH      27) NAVAJO         38) UCT
  6* AUSTRALIA       17) HONGKONG       28) NZ-CHAT        39) UNIVERSAL
  7* BRAZIL          18) ICELAND        29) NZ             40* US
  8* CANADA          19* INDIAN         30* PACIFIC        41) UTC
  9) CET             20) IRAN           31) POLAND         42) W-SU
 10* CHILE           21) ISRAEL         32) PRC            43) WET
 11) CUBA            22) JAMAICA        33) ROC            44) ZULU

Press "Return" to redisplay, enter "=" to search or "?" for help, or
Select the number above that best represents the desired time zone: nn

US Time Zone Menu                                         "*" indicates a menu

  0* RETURN TO MAIN TIME ZONE MENU
  1) ALASKA           4) CENTRAL          7) HAWAII          10) MOUNTAIN
  2) ALEUTIAN         5) EAST-INDIANA     8) INDIANA-STARKE  11) PACIFIC
  3) ARIZONA          6) EASTERN          9) MICHIGAN        12) SAMOA

Press "Return" to redisplay, enter "=" to search or "?" for help, or
Select the number above that best represents the desired time zone: n

You selected EASTERN / US as your time zone.
Is this correct? (Yes/No) [YES]:

    Configuring the Time Differential Factor (TDF)

    Default Time Differential Factor for standard time is -5:00.
    Default Time Differential Factor for daylight saving time is -4:00.

    The Time Differential Factor (TDF) is the difference between your
    system time and Coordinated Universal Time (UTC).  UTC is similar
```

```
    in most respects to Greenwich Mean Time (GMT).

    The TDF is expressed as hours and minutes, and should be entered
    in the hh:mm format.  TDFs for the Americas will be negative
    (-3:00, -4:00, etc.); TDFs for Europe, Africa, Asia and Australia
    will be positive (1:00, 2:00, etc.).

This time zone supports daylight saving time.
Is this time zone currently on daylight saving time? (Yes/No): n

Enter the Time Differential Factor [-5:00]:

    NEW SYSTEM TIME DIFFERENTIAL FACTOR = -5:00

Is this correct? [Y]:

    If you have Product Authorization Keys (PAKs) to register,
    you can register them now.

Do you want to register any Product Authorization Keys? (Yes/No) [Yes] n

    You can install the following products along with the OpenVMS
    operating system:

        o KERBEROS for OpenVMS I64 (required part of OpenVMS)
        o DECwindows Motif for OpenVMS I64
        o DECnet-Plus for OpenVMS I64
        o DECnet Phase IV for OpenVMS I64
        o HP TCP/IP Services for OpenVMS

    If you want to change your selections, you can do so later in the
    installation by answering "NO" to the following question:

        "Do you want the defaults for all options?"

    Do you want to install DECwindows Motif for OpenVMS I64 T1.4-1?
(Yes/No) [YES]

    Beginning with OpenVMS V7.1, the DECnet-Plus kit is provided with
    the OpenVMS operating system kit.  HP strongly recommends that
```

```
    DECnet users install DECnet-Plus.  DECnet Phase IV applications are
    supported by DECnet-Plus.

    DECnet Phase IV is also provided as an option.  Support for DECnet
    Phase IV is available through a Prior Version Support Contract.

    If you install DECnet-Plus and TCP/IP you can run DECnet
    applications over a TCP/IP network.  Please see the OpenVMS
    Management Guide for information on running DECnet over TCP/IP.

    Do you want to install DECnet-Plus for OpenVMS I64 T8.1?
(Yes/No) [YES]


    Do you want to install HP TCP/IP Services for OpenVMS T5.5-3T?
(Yes/No) [YES]

    The installation operation can provide brief or detailed descriptions.
    In either case, you can request the detailed descriptions by typing "?".

Do you always want detailed descriptions? (Yes/No) [No]

The following product has been selected:
    HP I64VMS OPENVMS V8.1                 Platform (product suite)

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for
any products that may be installed to satisfy software dependency requirements.

HP I64VMS OPENVMS V8.1: OPENVMS and related products Platform

    COPYRIGHT 1976,  8-DEC-2003

    Hewlett-Packard Development Company, L.P.


Do you want the defaults for all options? [YES]

    KERBEROS for OpenVMS I64 (required part of OpenVMS)
```

```
      HP I64VMS DWMOTIF T1.4-1: DECwindows Motif

        If a Local Language Variant is installed,refer to the Installation Guide.

        Do you want to continue? [YES]

Do you want to review the options? [NO]

Execution phase starting ...

The following products will be installed to destinations:
    HP I64VMS DECNET_PLUS T8.1             DISK$FBR6_V81:[VMS$COMMON.]
    HP I64VMS DWMOTIF T1.4-1               DISK$FBR6_V81:[VMS$COMMON.]
    HP I64VMS KERBEROS V2.0-14             DISK$FBR6_V81:[VMS$COMMON.]
    HP I64VMS OPENVMS V8.1                 DISK$FBR6_V81:[VMS$COMMON.]
    HP I64VMS TCPIP T5.5-3T                DISK$FBR6_V81:[VMS$COMMON.]
    HP I64VMS VMS V8.1                     DISK$FBR6_V81:[VMS$COMMON.]

Portion done: 0%...10%...20%...30%...40%...50%....60%...70%...80%...90%
%PCSI-I-PRCOUTPUT, output from subprocess follows ...
% - Execute SYS$MANAGER:TCPIP$CONFIG.COM to proceed with configuration of
%   HP TCP/IP Services for OpenVMS.
%
Portion done: 100%

The following products have been installed:
    HP I64VMS DECNET_PLUS T8.1             Layered Product
    HP I64VMS DWMOTIF T1.4-1               Layered Product
    HP I64VMS KERBEROS V2.0-14             Layered Product
    HP I64VMS OPENVMS V8.1                 Platform (product suite)
    HP I64VMS TCPIP T5.5-3T                Layered Product
    HP I64VMS VMS V8.1                     Operating System

HP I64VMS OPENVMS V8.1: OPENVMS and related products Platform

    HP I64VMS KERBEROS V2.0-14

      Configure the OpenVMS Kerberos clients & servers
```

```
  If Kerberos will be in use on this system
  and a current Kerberos configuration will
  not be used, please take the time to run
  the following command after the installation
  has completed (and after rebooting the system
  has completed (and after rebooting the system
  if this is an OpenVMS installation or upgrade):

      @SYS$STARTUP:KRB$CONFIGURE.COM

  After configuration, two system files need to
  be modified.   The following line should be
  added to SYS$MANAGER:SYSTARTUP_VMS.COM:

      $ @SYS$STARTUP:KRB$STARTUP

  The following line must be added to
  SYS$MANAGER:SYLOGIN.COM:

      $ @SYS$MANAGER:KRB$SYMBOLS

  The Kerberos 5 V2.0 documentation has been
  provided as it was received from MIT.  This
  documentation may differ slightly from the
  OpenVMS Kerberos implementation as it describes
  the Kerberos implementation in a Unix environment.
  The documents are:

      KRB$ROOT:[DOC]IMPLEMENT.PDF
      KRB$ROOT:[DOC]LIBRARY.PDF
      KRB$ROOT:[DOC]ADMIN-GUIDE.PS
      KRB$ROOT:[DOC]INSTALL-GUIDE.PS
      KRB$ROOT:[DOC]KRB425-GUIDE.PS
      KRB$ROOT:[DOC]USER-GUIDE.PS

HP I64VMS DWMOTIF T1.4-1: DECwindows Motif

  System reboot is required.

  If using a language variant, reboot after upgrade of language variant.
```

```
      Installation Verification Procedure can be run after reboot.

    HP I64VMS TCPIP T5.5-3T: HP TCP/IP Services for OpenVMS.

      Check the release notes for current status of the product.


    The installation is now complete.

    When the newly installed system is first booted, a special
    startup procedure will be run.  This procedure will:

    When the newly installed system is first booted, a special
    startup procedure will be run.  This procedure will:


        o  Configure the system for standalone or OpenVMS Cluster operation.
        o  Run AUTOGEN to set system parameters.
        o  Reboot the system with the newly set parameters.


    You may shut down now or continue with other operations.


  Process I64VMS_INSTALL logged out at 11-DEC-2003 11:34:50.59

Press Return to continue...

    ****************************************************************

    You can install or upgrade the OpenVMS I64 operating system
    or you can install or upgrade layered products that are included
    on the OpenVMS I64 operating system CD-ROM.

    You can also execute DCL commands and procedures to perform
    "standalone" tasks, such as backing up the system disk.

    Please choose one of the following:
```

```
        1)  Upgrade, install or reconfigure OpenVMS I64 Version V8.1
        2)  Display products and patches that this procedure can install
        3)  Install or upgrade layered products and patches
        4)  Show installed products
        5)  Reconfigure installed products
        6)  Remove installed products
        7)  Execute DCL commands and procedures
        8)  Shut down this system

Enter CHOICE or ? for help: (1/2/3/4/5/6/7/8/?) 8
       Shutting down the system
               SYSTEM SHUTDOWN COMPLETE


**** Primary HALTED with code HWRPB_HALT$K_REMAIN_HALTED
****   Hit any key to cold reboot   ****
P00>>>
```

B Linker Maps

This appendix provides the following seven Linker maps that illustrate changes to the Linker map files for this release:

- Object and Image Synopsis
- Image Segment Synopsis
- Program Segment Synopsis
- Symbol Cross Reference
- Symbols by Value
- Image Synopsis
- Link Run Statistics

The callout descriptions follow after the Linker maps.

Figure B-1 Linker Map 1

```
                                +-----------------------------+
                                ! Object and Image Synopsis ! (1)
                                +-----------------------------+

Module/Image (2)  File          Ident          Search (3) Lnkg (3) RFP (3)   Bytes  Creation Date      Creator
------------      ----          -----          ------     ----     ---     -----  ----------------   -------
GETJPI            V1.0          V1.0           Yes        No       No        832  19-NOV-2003 08:54  hp C X6.6-354
                  SYS$SYSROOT:[SYSMGR]GETJPI.OBJ;14
DECC$SHR          X8.1-00       X8.1-00        Yes                 No          0  3-NOV-2003 14:35   Linker T01-54
                  SYS$COMMON:[SYSLIB]DECC$SHR.EXE;1
SYS$PUBLIC_VECTORS X-2          Select         Yes                 No          0  3-NOV-2003 14:34   Linker T01-54
                  SYS$COMMON:[SYSLIB]SYS$PUBLIC_VECTORS.EXE;1


                                +-----------------------+
                                ! Cluster Synopsis ! (4)
                                +-----------------------+

Cluster (5)       Match (6)     Majorid   Minorid (6)
-------           -----         -------   ---------
MYCLU
DEFAULT_CLUSTER
DECC$SHR_         LESS/EQUAL        1           1
SYS$PUBLIC_VECTORS EQUAL            8837  268614460
```

Figure B-2 Linker Map 2

```
                +-------------------------+
                ! Image Segment Synopsis ! ❼
                +-------------------------+

Seg#  Cluster          Type    Pglts   Base Addr  Disk VBN  PFC  Protection  Attributes
----  -------          ----    -----   ---------  --------  ---  ----------  ----------
   0  MYCLU            LOAD        1   00010000         2    0  READ WRITE
   1                   LOAD        1   00020000         0    0  READ WRITE  DEMAND ZERO
   2                   LOAD        2   00030000         3    0  READ ONLY   EXECUTABLE
   3                   LOAD        1   00040000         5    0  READ ONLY
   4                   LOAD        1   00050000         6    0  READ ONLY
   5  DEFAULT_CLUSTER  LOAD        1   00060000         7    0  READ ONLY   SHORT
   6                   DYNAMIC     3   Q-00000000
                                     ❽ 80000000         8    0  READ ONLY

                Key for special characters above
                +-----------------------+
                !   Q  - Quadword       !
                +-----------------------+
```

Figure B-3 Linker Map 3

```
                                +-----------------------------+
                                ! Program Section Synopsis !❾
                                +-----------------------------+

Psect Name      Module/Image      Base     End      Length          Align       Attributes
----------      ------------      ----     ---      ------          -----       ----------

ITMLST                            00010000 0001000F 00000010 (      16.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,  MOD
                GETJPI            00010000 0001000F 00000010 (      16.) OCTA  4

FILLEN                            00020000 00020003 00000004 (       4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                <Linker>          00020000 00020003 00000004 (       4.) OCTA  4

FILLM                             00020010 00020013 00000004 (       4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                <Linker>          00020010 00020013 00000004 (       4.) OCTA  4

IOSB                              00020020 00020027 00000008 (       8.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                <Linker>          00020020 00020027 00000008 (       8.) OCTA  4

STATUS                            00020030 00020033 00000004 (       4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                <Linker>          00020030 00020033 00000004 (       4.) OCTA  4

$CODE$                            00030000 0003029F 000002A0 (     672.) OCTA  4 CON,REL,LCL,  SHR,  EXE,NOWRT,NOVEC,  MOD
                GETJPI            00030000 0003029F 000002A0 (     672.) OCTA  4

$LINKER_PLT_08_00                 000302A0 0003039F 00000100 (     256.) OCTA  4 CON,REL,LCL,  SHR,  EXE,NOWRT,NOVEC,  MOD
                <Linker>          000302A0 0003039F 00000100 (     256.) OCTA  4

$LITERAL$                         00040000 00040017 00000018 (      24.) OCTA  4 CON,REL,LCL,  SHR,NOEXE,NOWRT,NOVEC,  MOD
                GETJPI            00040000 00040017 00000018 (      24.) OCTA  4

$LINK$                            00050000 00050000 00000000 (       0.) OCTA  4 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,NOMOD
                GETJPI            00050000 00050000 00000000 (       0.) OCTA  4

$LINKER_UNWIND$                   00050000 0005002F 00000030 (      48.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
                GETJPI            00050000 0005002F 00000030 (      48.) QUAD  3

$LINKER_UNWINFO$                  00050030 00050077 00000048 (      72.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
                GETJPI            00050030 00050077 00000048 (      72.) QUAD  3

$LINKER_SDATA$                    00060000 000600E7 000000E8 (     232.) OCTA  4 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD,SHORT
                <Linker>          00060000 000600E7 000000E8 (     232.) OCTA  4
```

Figure B-4 Linker Map 4

```
                                     +------------------------+
                                     ! Symbol Cross Reference !
                                     +------------------------+

Symbol                          Value             Defined By                Referenced By ...
------                          -----             ----------                -----------------
DECC$$SHELL_HANDLER             000003A2-X        DECC$SHR                  GETJPI
DECC$EXIT                       00000200-X        DECC$SHR                  GETJPI
DECC$MAIN                       0000007B-X (10)   DECC$SHR                  GETJPI
DECC$TXPRINTF                   00000496-X        DECC$SHR                  GETJPI
ELF$TFRADR                      000600A0-R        GETJPI
FILLEN                          00020000-R        GETJPI                    GETJPI
FILLM                           00020010-R        GETJPI                    GETJPI
IOSB                            00020020-R        GETJPI                    GETJPI
ITMLST                          00010000-R        GETJPI
MAIN                            000301B0-RC       GETJPI
STATUS                          00020030-R        GETJPI                    GETJPI
SYS$GETJPIW                     0000009A-X        SYS$PUBLIC_VECTORS        GETJPI
SYS$IEEE_SET_FP_CONTROL         0000014E-X        SYS$PUBLIC_VECTORS        GETJPI
SYS$IEEE_SET_PRECISION_MODE     0000025D-X        SYS$PUBLIC_VECTORS        GETJPI
SYS$IEEE_SET_ROUNDING_MODE      0000025C-X        SYS$PUBLIC_VECTORS        GETJPI
__MAIN                          00030000-RC       GETJPI
```

Figure B-5 Linker Map 5

```
                                    +--------------------+
                                    ! Symbols By Value   !
                                    +--------------------+

Value       Symbols...
-----       ----------
00000078    X-DECC$MAIN
00000098    X-SYS$GETJPIW
0000014E    X-SYS$IEEE_SET_FP_CONTROL
00000200    X-DECC$EXIT
0000025C    X-SYS$IEEE_SET_ROUNDING_MODE
0000025D    X-SYS$IEEE_SET_PRECISION_MODE
000003A2    X-DECC$$SHELL_HANDLER
00000496    X-DECC$TXPRINTF
00010000    R-ITMLST
00020000    R-FILLEN
00020010    R-FILLM
00020020    R-IOSB
00020030    R-STATUS
00030000    RC-__MAIN
000301B0    RC-MAIN
000600A0    R-ELF$TFRADR

      Key for special characters above (11)

+-----------------------+
!   *  - Undefined      !
! (U) - Universal       !
!   R  - Relocatable    !
!   X  - External       !
!   C  - Code Address   !
!  WK  - Weak           !
! UxWk - Unix-Weak      !
+-----------------------+
```

Figure B-6 Linker Map 6

```
                                +-----------------+
                                ! Image Synopsis !
                                +-----------------+

Virtual memory allocated:       00010000 0006FFFF 00060000 (393216. bytes, 768. pages)
64-Bit Virtual memory allocated: 00000000 00000000 00000000
                                80000000 80010000 00010000 (65536. bytes, 128. pages)

Stack size:                          20. pages
Image header virtual block limits:    1.       1. (    1. block)
Image binary virtual block limits:    2.      10. (    9. blocks)
Image name and identification:  GETJPI V1.0
Number of files:                      5.
Number of modules:                    3.
Number of program sections:           7.
Number of global symbols:          3270.
Number of cross references:          36.
Number of image segments:             7.
User transfer address:          00030000
Number of code references to shareable images:
                                   3262.
Image type:                     EXECUTABLE.
Map format:                     FULL WITH CROSS REFERENCE in file SYS$SYSROOT:[SYSMGR]GETJPI.MAP;3
Estimated map length:           430. blocks
```

Figure B-7 Linker Map 7

```
                                +-----------------------+
                                ! Link Run Statistics !
                                +-----------------------+

Performance Indicators                Page Faults   CPU Time   Elapsed Time
----------------------                -----------   --------   ------------
  Command processing:                          41   00:00:00.04   00:00:00.03
  Pass 1:                                     149   00:00:00.22   00:00:00.21
  Allocation/Relocation:                        8   00:00:00.03   00:00:00.06
  Pass 2:                                       9   00:00:00.02   00:00:00.07
  Symbol table output:                          0   00:00:00.00   00:00:00.05
    Map data after object module synopsis:      4   00:00:00.01   00:00:00.01
Total run values:                             211   00:00:00.32   00:00:00.47

Quota usage ❷                    ByteCount  FileCount  PgFlCount
-------------                    ---------  ---------  ---------
  Available:                         65216        100      50000
  Command processing:                  576          4       8928
  Pass 1:                             1152          7      11072
  Allocation/Relocation:              1344          8      19392
  Pass 2:                             1536          9      19456
  Symbol table output:                1344          8      19456
    Map data after object module synopsis:    1344          8      19456

Using a working set limited to 16096 pages and 2381 pages of data storage (excluding image)

Number of modules extracted explicitly          = 0
  with 0 extracted to resolve undefined symbols
```

[1] The Alpha section titled, Object Module Synopsis. has been renamed on I64 to Object and Image Synopsis.

[2] The Alpha column Module Name has been renamed on I64 to Module/Image.

[3] New information has been added in columns titled, Search and Lnkg Info. Search is set to selective when the object or image was searched selectively. Lnkg is marked Yes when the object module contained linkage information.RFP is marked Yes when the object module was compiled with a Reduced Floating-Point model. This field is suppressed for shareable images.

[4] The Alpha section titled, Image Section Synopsis, has been divided into two sections for OpenVMS I64: Cluster Synopsis and Image Segment Synopsis. The Cluster Synopsis section no longer contains image sections, which on OpenVMS I64 are referred to as *segments*. Further, image sections are no longer printed for images, which was formerly done on VAX map files whenever there was a possibility of having based shareable images. (Based on shareable images are not allowed on OpenVMS Alpha or OpenVMS I64 systems.

[5] The Cluster column shows clusters that were created for and used by the linker and the order in which they were processed.

[6] The Match and Majorid/Minorid colulmns show version criteria, if any.

[7] The Image Segment Synopsis section shows each image segment as it was created. It contains the remaining columns of the Image Section Synopsis. The first column, Seg #, contains the image segment's number, which is used in the relocations that are applied to it. (See an analysis of an image for a display of the segment number in relocations.) The OpenVMS Alpha section, Protection and Paging, has been divided into two columns, Protection and Attributes, on OpenVMS I64. The column, Global Section Name, was eliminated.

[8] The dynamic segment is now mapped by default to P2 space.

[9] The section attributes PIC and NOPIC are not valid attributes on I64 and were removed.

[10] The designation of an external symbol has changed from Alpha maps. The prefix or suffix used on Alpha was **RX**, meaning relocatable and external. But the linker does not know whether an external symbol is relocatable or not. So, on I64 systems, the prefix or suffix has been changed to **X** (external).

[11] The Keys for Special Characters have been changed as follows:

- On I64, the symbol **C** appears for code address. When function does not have a function descriptor assigned by the linker, its value is its code address.
- On Alpha, a universal symbol appeared once along with its internal value. The map never displayed the external, universal value of the symbol. For example:

  ```
  D                               00020010-RU
  ```

  The Universal symbols on I64 now appear as:

  ```
  D                               00010010-R              FOO                             FOO
  D (U)                           00000001                <Linker Option>
  ```

- **D** appears twice, once with its internal value (00010010-R) and once with its external, universal value (00000001, its symbol vector index).
- The Alpha prefix or suffix **A** and **I** denote the difference between the alias name and the internal name for a symbol declared with the symbol vector clause as follows:
- On Alpha, the name with the **A** suffix or prefix would be the alias name and would contain the symbols external value – its symbol vector offset. The name with the **I** suffix or prefix would contain the images internal value – R for relocatable, and so on. For example:

  ```
  A                               00020000-RI                 FOO                 FOO
  A_ALIAS                         00000000-RUA                                    FOO
  ```

- **A** is replaced by **(U)** on I64. The universal alias name appears with a **(U)** and contains a symbol vector index. It does not appear on I64, and the symbol is displayed with its internal value.

  ```
  A                               00000000-R                  FOO                 FOO
  A_ALIAS (U)                     00000000                    <Linker Option>
  ```

- As described in callout 10, the designation for an external symbol is now **X** (external).

  ```
  DECC$MAIN             0000007B-X                            DECC$SHR            FOO
  ```

- UNIX weak symbols, designated with UxWk, are a new addition to OpenVMS. They are similar to OpenVMS weak symbols; however, more than one symbol with a UNIX weak definition can be processed when linking multiple modules without producing a multiple definitions error. UNIX weak symbols are currently produced by the C++ compiler.

```
UnixWeakSym             00000080-RUxWk                          FOO
```

[12] A new section titled, Quota Usage, was added to the Link Run Statistics section to keep track of the quotas that are being used by the I64 linker. FileCount keeps track of how many files the I64 linker has open. On Alpha, the linker kept the number of files open less than or equal to the file limit. Note, an error will occur on an I64 system if the file limit is exceeded.

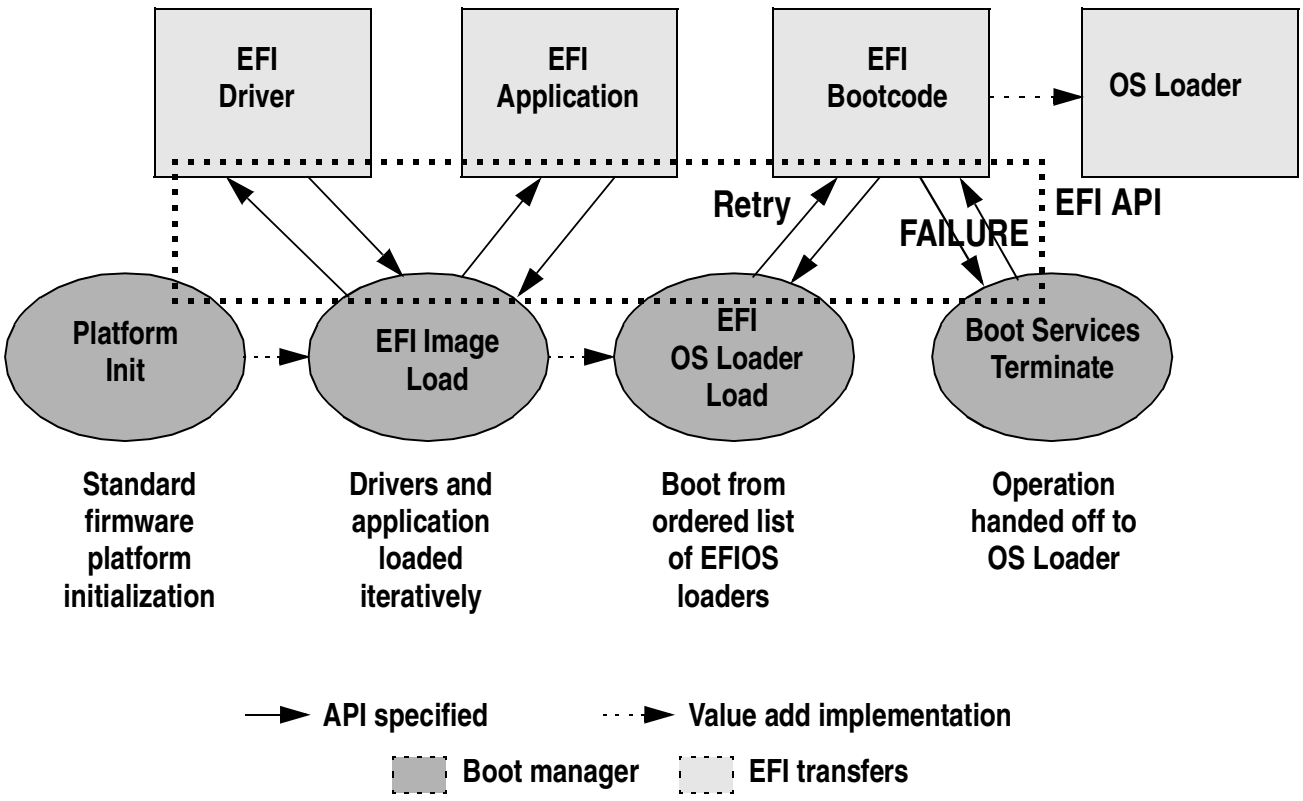
C Extensible Firmware Interface (EFI) Boot Manager

EFI (Extensible Firmware Interface) is an OS and platform-independent boot and pre-boot interface. EFI lies between the OS and platform firmware, allowing the OS to boot without having details about the underlying hardware and firmware. EFI supports boot devices; uses a flat memory model; and hides platform and firmware details from the OS.

NOTE EFI and Pre-OS System Environment (POSSE) are similar. EFI is an Intel specification, whereas POSSE is the HP implementation that aids HP support.

EFI supports booting from media that contain an EFI OS loader or an EFI-defined System Partition. An EFI-defined System Partition is required by EFI to boot from a block device.

Figure C-1 EFI Boot Sequence



Extensible Firmware Interface (EFI) Boot Manager

The EFI boot manager loads EFI applications (including OS first stage loader) and EFI drivers from an EFI-defined file system or image loading service. NVRAM variables point to the file to be loaded. These variables contain application-specific data that is passed directly to the EFI application. EFI variables provides system firmware a boot menu that points to all the operating systems, even multiple versions of the same operating systems.

The EFI boot manager allows you to control the server's booting environment. Depending on how you have configured the boot options, after the server is powered up the boot manager presents you with different ways to bring up the system. For example, you can boot to the EFI shell, to an operating system located on the network or residing on media in the server, or the Boot Maintenance menu.

- *Boot from a File*—Automatically adds EFI applications as boot options or allows you to boot from a specific file. When you choose this option, the system searches for an EFI directory. If the EFI directory is found, then it looks in each of the subdirectories below EFI. In each of those subdirectories, it looks for the first file that is an executable EFI application. Each of the EFI applications that meet this criterion can be automatically added as a boot option. In addition, legacy boot options for A: and C: are also added if those devices are present. You can also launch a specific application without adding it as a boot option. In this case the EFI boot manager searches the root directories and the \EFI\TOOLS directories of all of the EFI system partitions present in the system for the specified EFI application.
- *Add a Boot Option*—Adds a boot option to the EFI boot manager. You specify the option by providing the name of the EFI application. Along with the name, you can also provide either ASCII or UNICODE arguments the file might use. Given the EFI application name and any options, the EFI boot manager searches for the executable file in the same directories as described in "Boot from a File" option. When the file is found, it is executed.
- *Delete Boot Options*—Deletes a specific boot option or all boot options
- *Change Boot Order*—Controls the relative order in which the EFI boot manager attempts boot options. For help on the control key sequences you need for this option, refer to the help menu.
- *Manage BootNext Setting*—Selects a boot option to use one time (the next boot operation)
- *Set Automatic Boot Timeout*—Defines the value in seconds that pass before the system automatically boots without user intervention. Setting this value to zero disables the timeout feature.
- *Exit*—Returns control to the EFI boot manager main menu. This displays the active boot devices, including a possible integrated shell (if the implementation is so constructed).

EFI Commands

Table C-1 lists EFI commands.

Table C-1 EFI Commands

EFI Shell Command	Definition
help <command>	Display help for specified command or menu
reset	Reset the system
exit (at EFI shell)	Return to the main menu
EFI boot manager “change boot order”	Display or modify a path
bcfg	Search for boot devices
Many commands offer a [-b] parameter to cause 25 line breaks	Display or change scrolling capability
Configuration	
autoboot	Display or set the auto start flag
info boot	Display or set processor boot identifier
EFI boot manager	Display boot-related information
autoboot	Seconds allowed for boot attempt
cpuconfig	Config/deconfig processor
boottest	Display or set boot tests execution
date	Read or set the date
time	Read or set the real time clock
Information	
info all	Display all system information
info boot	Display boot-releated information
info cpu	Display cache information
info chiprev	Display revision number of major VLSI
MP command <df>	Display FRU information
info fw	Display firmware version for PDC, ICM, and complex
info io	Display firmware version for PDC, ICM, and complex
LanAddress	Display core LAN station address
info mem	Display memory information
info cpu	Display processor information

Table C-1 EFI Commands (Continued)

EFI Shell Command	Definition
Service	
errdump clear	Clear (zero) the contents of PIM
mm	Read memory locations scope of page deallocation
PDT	Display or clear the page deallocation table
errdump mca errdump cmc errdump init	Display PIM information (processor internal memory)

EFI/POSSE Commands

This section describes the EFI/POSSE commands developed for the server.

NOTE EFI and Pre-OS System Environment (POSSE) are similar. EFI is an Intel® specification, whereas POSSE is the HP implementation that aids HP support.

help

Provides information on the EFI shell commands.

Syntax

```
help [-b] <category>
help [-b] <cxnd>
help [-b] bch <bchmenu> <bchcmd>
```

Parameters

```
-b              Enable page breaking
category        Category of commands to view help on commands
bch             Display the list of SCM commands and their corresponding EFI bchmenu BCH
                menu name taken from the top level of the BCH menu bchcmd BCH command on
                  which to display information.
```

Operation

If help is invoked with no parameters, it displays a list of shell command categories. To list all of the commands within a category, the user should type **`help <category>`** (see examples). If invoked with the **`-b`** switch, any output longer than one page pauses after each page is displayed. If a shell command name is used as a parameter, verbose help is displayed for that command.

If help is invoked with the `bch` option, it displays a list of BCH commands and their corresponding EFI/POSSE commands. It instructs the user to repeat the command line followed by a menu name for more information on that menu. If help is invoked followed by **`bch`** and a menu name, it displays a list of commands that appear under that BCH menu. The user may then invoke help followed by **`bch`**, the menu name, and a BCH command name to display information on that command. This would point the user to the command that has taken the place of that BCH functionality, or will inform the user that the functionality no longer exists. As a shortcut, the user may enter help followed by bch and a BCH command name to go straight to that command.

Example C-1 `help` Command

```
Shell> help
List of classes of commands:

boot          -- Booting options and disk-related commands
configuration -- Changing and retrieving system information
devices       -- Getting device, driver and handle information
memory        -- Memory related commands
shell         -- Basic shell navigation and customization
```

```
scripts       -- EFI shell-script commandsType "help" followed by a class name for a list of
commands in that class
Type "help" followed by command name for full documentation
```

Example C-2 `help configuration` Command

```
Shell> help configuration
Configuration commands:

cpuconfig  -- Deconfigure or reconfigure cpus
date       -- Display or set date
err        -- Display or set error level
esiproc    -- Make an ESI call
errdump    -- View/Clear logs
info       -- Display hardware information
monarch    -- View or set the monarch processor
palproc    -- Make a PAL call
salproc    -- Make a SAL call
time       -- Display or set time
ver        -- Displays version info

Type "help" followed by command name for full documentationon that command.
Type "help -a" to display a list of all commands.
```

Example C-3 `help cpuconfig` Command

```
Shell> help cpuconfig

CPUCONFIG [cpu] [on|off]

   cpu    Specifies which cpu to configure
   on|off Specifies to configure or deconfigure a cpu

   Notes:
        1. Cpu status will not change until next boot

   Examples:
       * To deconfigure CPU 0
       fs0:\> cpuconfig 0 off
       CPU will be deconfigured on the next boot

       * To display configuration status of cpus
       fs0:\> cpuconfig
       <CPU configuration data displayed>
```

Example C-4 help bch Command

```
    COnfiguration         help bch co
     INformation           help bch in
     PAth                  help bch pa
     ScRool                help bch sr
     SEArch                help bch sea
     SERvice               help bch ser
     BOot                  help bch bo
     HElp                  help bch he
     RESET                 help bch reset
     MAin                  help bch ma
```

```
    For more help on one of the commands above, at the prompt type:
        help bch COMMAND
```

baud

Sets the baud rate and communication settings for a UART.

Syntax

```
baud index baudrate
```

Parameters

```
index         0 through the total number of UARTS minus one
baudrate      baud rate.
```

Operation

This command is used to change the speed for a UART in the system. This command works for all UARTs visible to EFI/POSSE. If the UART is part of PDH space and is initialized by the core firmware, this command communicates the settings to core firmware so the UART can be initialized with the new settings on the next boot. System default is 9600 baud.

Other Communication parameters are listed in Table C-2.

Table C-2 Communications Parameters

Parameter	Value
RECEIVE_FIFO_DEPTH	1
TIMEOUT	1000000
PARITY	No parity
DATA_BITS	8
STOP_BITS	1
CONTROL_MASK	0

boottest

Interacts with the speedy boot variable allowing it to be set appropriately.

Syntax

```
boottest                        Displays status of all speedy boot bits
boottest on                     Run all tests (for a normal boot time)
boottest off                    Skip all tests (for a faster boot time)
boottest [test]                 Displays status of specific Speedy Boot bit
boottest [test] [on|off]        Sets or clears a specific Speedy Boot bit
```

Parameters

```
[test]  Each test can be set or cleared:
        booting_valid   Enable/disable system firmware response to BOOTING
                        bit. If OS Speedy Boot aware set to on.
```

```
       early_cpu       Enable/disable early CPU selftests.
        late_cpu        Enable/disable late CPU selftests.
        platform        Enable/disable system board hardware tests.
        chipset         Enable/disable CEC tests.
```

Example C-5 `boottest` Command

```
Shell> boottest
   BOOTTEST Settings Default Variable
   Selftest         Setting
   -------------    -------------------------
   early_cpu        Run this test
   late_cpu         Run this test
   platform         Run this test
   chipset          Run this test
   io_hw            Run this test
   mem_init         Run this test
   mem_test         Run this test
```

clearlogs

Clearlogs clears the SEL and FPL logs. This command is not documented in the online help.

clearlogs command

```
Shell> clearlogs

Clear SEL and FPL logs? [y.n] y

FPL cleared

SEL cleared
```

cpuconfig

Displays the CONFIG/DECONFIG state of processors in the system and allows the user to configure or reconfigure processors.

cpuconfig command

Syntax

```
cpuconfig cpu on|off
```

Parameters

```
cpu           specify a processor
on|off        state to which you set the processor
```

Example C-6 `cpuconfig` Command

```
Shell>  cpuconfig
PROCESSOR MODULE INFORMATION
       # of                  L3     L4     Family/
CPU    Logical               Cache  Cache    Model           Processor
Slot   CPUs       Speed      Size   Size    (hex.)     Rev     State
```

```
---   -------    ---------  ----   -----   ------   ----    ----------
 0         1       900 MHz   1.5MB   None   1F/00     B3          Active
 1         1       900 MHz   1.5MB   None   1F/00     B3          Active
```

default

Allows the user to restore NVM to default values and clear NVM storage values.

Syntax

```
default     [efi|sal]
default      clear [bmc|efi|sal]
```

Parameters

```
clear     clears NVM storage values
```

Operation

Default sets NVM and Stable Store values to predefined default values. To the normal user only a subset of values are available for default. Executing “default clear” resets the system.

errdump

Displays the contents of processor internal memory logged on the first MCA for all processors present in the system.

Syntax

```
errdump [mca | cpe | cmc | init | la | clear]
```

Parameter

```
mca               Dumps the Machine Check Abort error log.
cpe               Dumps the Corrected Platform Error log.
cmc               Dumps the Corrected Machine Check log.
init              Dumps the Initialization log.
la                Dumps the Logic Analyzer log.
clear             Erases all of the logs.
```

Operation

If a user enters no parameters, the usage is displayed. Otherwise, the specified error log is displayed. Adding -n to the clear parameter disables the confirmation prompt. (The `errdump` command can also be accessed by way of the System Configuration menu.)

info

Allows the user to display most system information.

Syntax

```
info [ -b] [target]
```

Parameters

```
target:            Valid targets are:
        all        display everything
        cpu        display information on cpus
        cache      display information on cache
        mem        display information on memory
        io         display information on io
        boot       display boot-related information
        chiprev    display information on chip revisions
        fw         display firmware version information
        sys        display system information
        warning    display warning and stop boot information
```

info all command

```
Shell> info all


SYSTEM INFORMATION


Date/Time:  Oct 28, 2003  18:07:32  (20:03:10:28:18:07:32)
Manufacturer:  hp
Product Name:  server rx2600
Product Number:  A6870A
Serial Number:  US30464638
UUID:  C198BA79-478A-11D7-9C22-6033AC66036B
System Bus Frequency:  200 MHz


PROCESSOR MODULE INFORMATION


         # of                L3      L4      Family/
   CPU   Logical             Cache   Cache   Model          Processor
   Slot  CPUs      Speed     Size    Size    (hex.)    Rev  State
   ----  -------  --------  ------  ------  -------  ---  ------------
     0      1      900 MHz  1.5 MB    None   1F/00    B3        Active
     1      1      900 MHz  1.5 MB    None   1F/00    B3        Active



MEMORY INFORMATION


     ---- DIMM A -----  ---- DIMM B -----
      DIMM   Current     DIMM   Current
```

```
---  ------ ----------  ------ ----------
 0    256MB     Active   256MB     Active
 1    256MB     Active   256MB     Active
 2    ----               ----
 3   ----        ----
 4    ----               ----
 5    ----               ----

   Active Memory    : 1024 MB
   Installed Memory : 1024 MB


I/O INFORMATION


BOOTABLE DEVICES

   Order  Media Type  Path
   -----  ----------  --------------------------------------
     1    HARDDRIVE   Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)/HD(Part1,SigC1)

   Seg  Bus  Dev  Fnc  Vendor  Device Slot
    #    #    #    #     ID      ID     #   Path
   ---  ---  ---  ---  ------  ------  ---  -----------
    00   00   01   00  0x1033  0x0035   XX  Acpi(HWP0002,0)/Pci(1|0)
    00   00   01   01  0x1033  0x0035   XX  Acpi(HWP0002,0)/Pci(1|1)
    00   00   01   02  0x1033  0x00E0   XX  Acpi(HWP0002,0)/Pci(1|2)
    00   00   02   00  0x1095  0x0649   XX  Acpi(HWP0002,0)/Pci(2|0)
    00   00   03   00  0x8086  0x1229   XX  Acpi(HWP0002,0)/Pci(3|0)
    00   20   01   00  0x1000  0x0030   XX  Acpi(HWP0002,100)/Pci(1|0)
    00   20   01   01  0x1000  0x0030   XX  Acpi(HWP0002,100)/Pci(1|1)
    00   20   02   00  0x14E4  0x1645   XX  Acpi(HWP0002,100)/Pci(2|0)
    00   40   01   00  0x1000  0x0021   03  Acpi(HWP0002,200)/Pci(1|0)
    00   40   01   01  0x1000  0x0021   03  Acpi(HWP0002,200)/Pci(1|1)
    00   E0   01   00  0x103C  0x1290   XX  Acpi(HWP0002,700)/Pci(1|0)
    00   E0   01   01  0x103C  0x1048   XX  Acpi(HWP0002,700)/Pci(1|1)
    00   E0   02   00  0x1002  0x5159   XX  Acpi(HWP0002,700)/Pci(2|0)
```

```
BOOT INFORMATION

   Monarch CPU:

  Current  Preferred
   Monarch   Monarch   Possible Warnings
   -------  ---------  -----------------
         0          0

   AutoBoot: ON - Timeout is : 10 sec

   Boottest:

   BOOTTEST Settings Default Variable

   OS is not speedy boot aware.


   Selftest     Setting
  ---------    --------------
   early_cpu    Skip this test
   late_cpu     Skip this test
   platform     Skip this test
   chipset      Skip this test
   io_hw        Skip this test
   mem_init     Skip this test
   mem_test     Skip this test

   LAN Address Information:

   LAN Address        Path
   -----------------  ---------------------------------------
   Mac(00306E4A133D)  Acpi(HWP0002,0)/Pci(3|0)/Mac(00306E4A133D))
  *Mac(00306E4A121F)  Acpi(HWP0002,100)/Pci(2|0)/Mac(00306E4A121F))

FIRMWARE INFORMATION

   Firmware Revision: 2.20 [4331]
```

```
   PAL_A Revision: 7.31/7.31
   PAL_B Revision: 7.59

  SAL Spec Revision: 3.01
   SAL_A Revision: 2.00
  SAL_B Revision: 2.20

   EFI Spec Revision: 1.10
   EFI Intel Drop Revision: 14.61
   EFI Build Revision: 1.22

   POSSE Revision: 0.10

   ACPI Revision: 7.00

   BMC Revision 1.50
   IPMI Revision: 1.00
   SMBIOS Revision: 2.3.2a
   Management Processor Revision: E.02.22


WARNING AND STOP BOOT INFORMATION


CHIP REVISION INFORMATION

   Chip                  Logical     Device       Chip
   Type                     ID         ID       Revision
  -------------------   -------     ------     --------
   Memory Controller         0       122b         0023
    Root Bridge               0       1229         0023

     Host Bridge          0000       122e         0032
     Host Bridge          0001       122e         0032
     Host Bridge          0002       122e         0032
     Host Bridge          0003       122e         0032
     Host Bridge          0004       122e         0032
     Host Bridge          0006       122e         0032
     Host Bridge          0007       122e         0032
```

```
     Other Bridge            0          0         0002
       Other Bridge          0          0         0007
         Baseboard MC        0          0         0150
```

lanaddress

Allows the user to display the core I/O MAC address.

Syntax:

```
lanaddress
```

Parameters

none

Example C-7 `lanaddress` Command

```
Shell> lanaddress

LAN Address  Information
    LAN ADDRESS         Path
    ------------------  -------------------------
     Mac(00306E4A133D)   Acpi(HWP0002,0)/Pci(3|0)/Mac(00306E4A133D))
     *Mac(00306E4a121f)  Acpi(HWP0002,100)/Pci(2|0)/Mac(00306E4a121f))
```

monarch

Displays or modifies the ID of the bootstrap processor. The preferred monarch number is stored in NVM.

Syntax

```
monarch cpu
```

Parameters

```
cpu      specifies a cpu
```

Operation

If specified with no parameters, **`monarch`** displays the Monarch processor for the system. Specifying a processor number alters the preferred Monarch processor. None of these changes takes affect until after a reboot.

Example C-8 `monarch` Command

```
Shell> monarch
 Current   Preferred
 Monarch    Monarch     Possible Warnings
 -------   ---------    -----------------
       0           0
```

pdt

Displays or clears the contents of the Page Deallocation Table.

Syntax

```
pdt (clear)
```

Parameters

```
clear      clears the pdt
```

Operation

With no options specified, the command displays the PDT information for the system. The PDT is cleared and a reboot is required for memory reallocation and safe booting.

Example C-9 `pdt` Command

```
Shell> pdt
PDT Information
                Last Clear time: PDT has not been cleared
       Number of total entries in PDT:                  50
        Number of used entries in PDT:                   0
        Number of free entries in PDT:                  50
  Number of single-bit entries in PDT:                   0
   Number of multi-bit entries in PDT:                   0
     Address of first multi-bit error:  x0000000000000000
```

Example C-10 `pdt clear` Command

```
Shell> pdt clear
Are you sure you want to clear the PDT? [y/N] y
Shell>

Shell> pdt
PDT Information

                Last Clear time: 10/21/01  5:00p
       Number of total entries in PDT:                  50
        Number of used entries in PDT:                   0
        Number of free entries in PDT:                  50
  Number of single-bit entries in PDT:                   0
   Number of multi-bit entries in PDT:                   0
     Address of first multi-bit error: 0x0000000000000000
```

sysmode

Display or modify the system mode.

Syntax

```
sysmode <normal | admin| service>
```

Parameters

```
<normal>     sets system mode to normal

<admin>      sets system mode to admin

<service>     sets system mode to service
```

Operation

If specified alone, sysmode displays the system mode. If a mode is specified as a parameter, then the system mode is changed. This new mode takes effect immediately. The system mode is retained on successive boots. Interaction with sysmode in a variety of scenarios is outlined below.

Example C-11 `sysmode` Command

```
Shell> sysmode
System Mode: NORMAL

Shell> sysmode admin
You are now in admin mode.

Shell> sysmode service
You are now in service mode.

Shell> sysmode normal
You are now in normal mode
```

Specifying SCSI Parameters

The following SCSI parameters may be configured for the SCSI board:

- SCSI ID (SCSI initiator ID)
- Maximum data transfer rate (SCSI rate)
- Bus width
- Whether the HBA is bootable (driver support)
- Avoid bus resets (secondary cluster server)

Using the SCSI Setup Utility

Step 1. At the EFI shell prompt, type this command to map the parameters for all PCI cards installed in the system:

`info io`

A list of all the devices that are installed in the HP Integrity rx2600 server and managed by EFI drivers is displayed. The output may look like this:

```
Shell> info io

I/O INFORMATION

BOOTABLE DEVICES

      Order       Media  Type    Path
       -------        -----------------
------------------------------------------------------------
          1            HARDDRIVE
Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)/HD(Part1,SigC1)

Seg  Bus  Dev  Fnc  Vendor  Device Slot
    #    #    #    #     ID      ID     #   Path
   ---  ---  ---  ---  ------  ------  ---  -----------
    00   00   01   00  0x1033  0x0035   XX  Acpi(HWP0002,0)/Pci(1|0)
    00   00   01   01  0x1033  0x0035   XX  Acpi(HWP0002,0)/Pci(1|1)
    00   00   01   02  0x1033  0x00E0   XX  Acpi(HWP0002,0)/Pci(1|2)
    00   00   02   00  0x1095  0x0649   XX  Acpi(HWP0002,0)/Pci(2|0)
    00   00   03   00  0x8086  0x1229   XX  Acpi(HWP0002,0)/Pci(3|0)
    00   20   01   00  0x1000  0x0030   XX  Acpi(HWP0002,100)/Pci(1|0)
    00   20   01   01  0x1000  0x0030   XX  Acpi(HWP0002,100)/Pci(1|1)
    00   20   02   00  0x14E4  0x1645   XX  Acpi(HWP0002,100)/Pci(2|0)
    00   40   01   00  0x1000  0x0021   03  Acpi(HWP0002,200)/Pci(1|0)
    00   40   01   01  0x1000  0x0021   03  Acpi(HWP0002,200)/Pci(1|1)
    00   E0   01   00  0x103C  0x1290   XX  Acpi(HWP0002,700)/Pci(1|0)
```

```
00   E0   01   01  0x103C  0x1048   XX  Acpi(HWP0002,700)/Pci(1|1)

00   E0   02   00  0x1002  0x5159   XX  Acpi(HWP0002,700)/Pci(2|0)
```

In the example above, *two* SCSI boards are in the listing. The information for *both* channels of *both* SCSI boards is shown in **bold**, for highlighting purposes.

For each channel of the SCSI board, you need to note certain information. As an example, look at the information for the first SCSI board (the first two bold lines). For each channel of *this* SCSI board, note the following information:

- `Bus #`—identifies the bus the device is on; for the SCSI board, this is the same for both channels. In this example, the bus number is `40`.
- `Dev #`—the ID the device is assigned on the bus; for the SCSI board, this is the same for both channels. In this example, the SCSI board is device `01`.
- `Fnc #`—identifies the channel of the device (`00` for channel A, `01` for channel B, and so on). In this example, because the SCSI board has two channels, one channel is `00` and the other is `01`.
- `Vendor ID`—shows the device's vendor ID; for the SCSI board, this is the same for both channels. For all SCSI board HBAs, the ID is `0x1000`.
- `Device ID`—shows the device's device ID; for the SCSI board, this is the same for both channels. For all SCSI board HBAs, the ID is `0x0021`.
- `Slot #`—identifies the physical card slot in the system where the HBA is installed; for the SCSI board, this is the same for both channels. In this example, the HBA is in slot `03`.
- `Path`—identifies the device's path; for the SCSI board, this is the same for both channels. In this example, the HBA's path is `Acpi(HWP0002,200)/Pci(1|0)` for channel A and `Acpi(HWP0002,200)/Pci(1|1)` for channel B.

Using the SCSI board's information from the example above, the pieces of information that, combined, tell you this is a SCSI board are the following (shown in **bold**, for highlighting purposes):

```
00   40   01   00  0x1000  0x0021   03  Acpi(HWP0002,200)/Pci(1|0)
00   40   01   01  0x1000  0x0021   03  Acpi(HWP0002,200)/Pci(1|1)
```

Looking at all of the above information together, the vendor (**0x1000**) and device (**0x0021**) are the IDs for a SCSI board. Of the devices with those IDs, this device has two channels (Fnc # of **00** immediately followed by Fnc # of **01**). Also, this SCSI board has a numeric (non-XX) slot # (**03**, in this example).

Step 2. Still at the EFI shell prompt, type this command to obtain the controller's handle for the SCSI card:

```
devtree
```

A tree of all EFI-capable devices installed in the system is displayed. The output could look like this:

```
Shell> devtree
Device Tree

 Ctrl[04]
  Ctrl[0A] Acpi(HWP0002,0)
    Ctrl[15] Usb Open Host Controller
      Ctrl[3D] Acpi(HWP0002,0)/Pci(1|0)/Usb(1, 0)
    Ctrl[16] Usb Open Host Controller
       Ctrl[17] Acpi(HWP0002,0)/Pci(1|2)
```

```
  Ctrl[18] PCI IDE/ATAPI Controller
     Ctrl[50] DV-28E-B
  Ctrl[19] Acpi(HWP0002,0)/Pci(3|0)
    Ctrl[51] Acpi(HWP0002,0)/Pci(3|0)/Mac(00306E4A133D)
Ctrl[0B] Acpi(HWP0002,100)
   Ctrl[1A] LSI Logic Ultra320 SCSI Controller
   Ctrl[1B] LSI Logic Ultra320 SCSI Controller
   Ctrl[1C] Acpi(HWP0002,100)/Pci(2|0)
      Ctrl[53] Broadcom NetXtreme Gigabit Ethernet Adapter
Ctrl[0C] Acpi(HWP0002,200)
      Ctrl[1D] LSI Logic Ultra160 SCSI Controller
      Ctrl[1E] LSI Logic Ultra160 SCSI Controller
Ctrl[0D] Acpi(HWP0002,300)
Ctrl[0E] Acpi(HWP0002,400)
Ctrl[0F] Acpi(HWP0002,600)
Ctrl[10] Acpi(HWP0002,700)
      Ctrl[1F] Acpi(HWP0002,700)/Pci(1|0)
      Ctrl[20] Acpi(HWP0002,700)/Pci(1|1)
         Ctrl[3B] 16550 Serial UART Driver
            Ctrl[3C] VT-100 Serial Console
               Ctrl[36] Primary Console Input Device
               Ctrl[37] Primary Console Output Device
               Ctrl[35] Primary Standard Error Device
      Ctrl[21] Acpi(HWP0002,700)/Pci(2|0)
 Ctrl[12] VenHw(B19EEDB4-BCC0-11D4-8046-0010B5481A73)
 Ctrl[38] Acpi(PNP0501,0)
      Ctrl[39] 16550 Serial UART Driver
          Ctrl[3A] VT-100 Serial Console
Ctrl[4B] VenHw(904EFCF0-F0A8-11D4-B4CA-303031303833)
Ctrl[4E] VenHw(D65A6B8C-71E5-4DF0-A909-F0D2992B5AA9)
```

In the above example, *this* SCSI board's information is shown in **bold**, for highlighting purposes. You can tell the information is for this SCSI board because the path on the first line—`Acpi(HWP0002,200)`— is the HBA's path from the information displayed by the `info io` command. The next two lines are for the SCSI board's two channels, one line for each channel (they contain the SCSI board's description [`LSI Logic Ultra160 SCSI Controller`]). Note the value shown for `Ctrl`—`82` and `83`—at the beginning of each of those lines; this is the **controller's handle** for that channel. You need to know it for the next step.

NOTE The controller's handle values will change on every boot.

Step 3. Still at the EFI shell prompt, type this command to obtain the EFI driver's handle for the SCSI card:

```
drvcfg
```

A list of all EFI-capable configurable components in the system is displayed. The output may look like this:

Shell> drvcfg
Configurable Components

```
Drv[43]  Ctrl[1D]  Lang[eng]
Drv[43]  Ctrl[1E]  Lang[eng]
Drv[44]  Ctrl[18]  Lang[eng]
```

```
Drv[46]   Ctrl[1C]   Lang[eng]
Drv[4C]   Ctrl[1A]   Lang[eng]
Drv[4C]   Ctrl[1B]   Lang[eng]
```

This listing shows which driver controls which device (controller). In the above example, *this* SCSI board's information is shown in **bold**, for highlighting purposes. You can tell the information is for this SCSI board because the values shown for `Ctrl`—`1D` and `1E` —are the controller's handles for the SCSI board's two channels (from the information displayed by the `devtree` command).

NOTE The EFI driver's handle values will change on every boot.

TIP From this command (`drvcfg`), we recommend you record these two pieces of information for *each* channel of *each* SCSI board HBA you want to change the SCSI parameters for:

- `Drv` (the EFI driver's handle)
- `Ctrl` (the controller's handle)

Step 4. Using the information (the driver's handle [`Drv`] and the controller's handle [`Ctrl`]) from the `drvcfg` command, start the EFI SCSI Setup Utility for *one* channel of *this* SCSI board. Still at the EFI shell prompt, type this command:

drvcfg -s dvr_handle cntrl_handle

where

- *`drvr_handle`* is the handle of the driver that controls the channel whose SCSI ID you want to display or change
- *`cntrl_handle`* is the handle of the controller for the channel whose SCSI ID you want to display or change

So, continuing the example for *channel A* of *this* SCSI board, you would type:

```
drvcfg -s 43 1E
```

Step 5. The EFI SCSI Setup Utility starts and its main menu is displayed, showing a list of all the EFI capable HBAs in the system.

TIP To move the cursor in the EFI SCSI Setup Utility, you can use these keys:

- Arrow keys: ↑ ↓ ← →
- Alternate keys:

  H = left

  J = down

  K = up

  L = right

  I = home

O = end

Move the cursor to highlight *this* channel of *this* SCSI board; press **Enter**. (To determine which channel of the HBA to highlight, match the `PCI Bus`, `PCI Dev`, and `PCI Func` values on this screen to the `Bus #`, `Dev #`, and `Fnc #` values from the `info io` command.)

CAUTION Do *not* select the `<Global Properties>` option on the main menu.

Step 6. The "Adapter Properties" screen for this channel of this SCSI board is displayed. If you like, you can make sure the utility is running for *this* channel of *this* SCSI board by comparing the values shown for `PCI Bus`, `PCI Device`, and `PCI Function` to the `Bus #`, `Dev #`, and `Fnc #` values from the `info io` command.

CAUTION Do *not* change the value for *any* of these fields on the "Adapter Properties" screen:

- `Auto Termination`
- `SCSI Parity`
- `SCSI Bus Scan Order`
- `Spinup Delay (Secs)`

Changing any of these fields can cause unpredictable results.

CAUTION Do *not* change the value for *any* of these fields on the "Device Properties" screen:

- `Scan Id`
- `Scan LUNs > 0`
- `Disconnect`
- `SCSI Timeout`
- `Queue Tags`
- `Format`
- `Verify`

Changing any of these fields can cause unpredictable results.

Step 7. You may display (and optionally change) any SCSI parameters listed below for *this* channel of *this* SCSI board, or restore its SCSI parameters to their default values.

- SCSI ID
- Maximum data transfer rate
- Bus width
- Whether the HBA is bootable (driver support)
- Avoid bus resets (secondary cluster server)

- Restore Defaults

Step 8. Use the arrow keys to navigate to the appropriate SCSI parameter.

Step 9. Use the plus (**+**) and minus (**-**) keys to scroll through the values until the value you want is displayed.

Step 10. Press **Esc** to exit the "Adapter Properties" screen. You are given these choices:

- Cancel the exit from the screen (to stay in the "Adapter Properties" screen for *this* channel of *this* SCSI board)
- Save the changes you made and then exit the screen
- Discard the changes you made and then exit the screen

Step 11. Move the cursor to the action (cancel, save, or discard) you want to take; press **Enter**.

If you selected cancel, you remain in the "Adapter Properties" screen for *this* channel of *this* SCSI board. You can still change *this* channel's parameters listed above.

If you selected save or discard, you are placed in the EFI SCSI Setup Utility's main menu.

CAUTION Do *not* select the `<Global Properties>` option on the main menu.

Step 12. Press **Esc** to exit the main menu and the EFI SCSI Setup Utility.

Step 13. Select the option for exiting the utility.

Step 14. When you are prompted to, press **Enter** to stop *this* SCSI board; you are now back at the EFI shell prompt.

Step 15. At the EFI shell prompt, type this command:

```
reset
```

The system starts to reboot. This is **required** to cause the new SCSI setting.

D Management Processor Commands

This appendix describes the management processor commands. The management processor provides a virtual front panel that can be used to monitor system status and see the state of front panel LEDs. All MP functions are available by way of the LAN, local RS-232 and remote RS-232 ports.

The management processor is available whenever the system is connected to a power source, even if the server main power switch is in the off position.

Access to the management processor can be restricted by user accounts. User accounts are password protected and provide a specific level of access to the server and management processor commands.

Multiple users can interact with the management processor. From the MP MAIN MENU, users can select any of the following options:

- Enter management processor command mode.
- Enter console.
- View event logs.
- View console history.
- Display virtual front panel.
- Enter console session.
- Connect to another management processor.

Multiple users can select different options from the MP MAIN MENU at the same time. However, management processor command mode and console mode are mirrored. The MP allows only one user at a time to have write access to the shared console.

MP Commands

Reset BMC Passwords

BP: Reset BMC Passwords

This command resets BMC passwords (both USER and ADMIN passwords.

Configure Serial Port Parameters

CA: Configure local and remote serial port parameters\

Set up the local serial port parameters as follows:

- TERMINAL TYPE: VT100 vs HPterm
- BAUD RATES: Input and output data rates are the same: 300, 1200, 2400, 4800, 9600, 38400, 115200 bit/sec.
- FLOW CONTROL: Hardware uses RTS/CTS; Software yses Xon/Xoff.
- TRANSMIT CONFIGURATION STRINGS: Disable this setting whenever the modem being used is not compatible with the supported modem (MT5634ZBA).

NOTE Important: Do not mix HP and VT100 terminal types at the same time.

Set up the remote serial port parameters as follows:

- MODEM PROTOCOL: Bell or CCITT (CCITT is a European standard; RTS/CTS signaling is used, as well as the Ring signal. Bell is a U.S. or simple mode).
- BAUD RATES: Input and output data rates are the same: 300, 1200, 2400, 4800, 9700, 38400, 115200 bit/sec.
- FLOW CONTROL: Hardware uses RTS/CTS; Software uses Xon/Xoff.
- TRANSMIT CONFIGURATION STRINGS: Disable this setting whenever the modem being used is not compatible with the supported modem (MT5634ZBA).
- MODEM PRESENCE: When the modem may not always be connected, set this parameter to “not always connected”.

For example: A modem attached through a switch. In mode “not always connected,” no dial-out functions are allowed: DIAL-BACK is disabled, and PAGING is not possible.

The MP mirrors the system console to the MP local, remote/modem, and LAN ports. One console output stream is reflected to all of the connected console users. If users use several different terminal types simultaneously, some users may see strange results.

Certificate Generate

CG: Generate RSA key pair or Self-Signed Certificate.

This command generates a new RSA key pair and self signed certificate.

Console Log

CL: Console Log—view the history of the Console output.

This command displays up to 60 Kilobytes of logged console data (about 60 pages of display in text mode) sent from the system to the Console path.

Command Mode

Command Mode

CM: Command Mode—enter command mode

This command switches the console terminal from the MP Main Menu to mirrored command interface mode. If the current console authority is Administrator and the new login is as an Operator, the command console is denied (remains in MP Main Menu mode). If a command is in progress, a message is displayed warning the new user of system status.

Console

CO: Console—leave command mode and enter console mode.

This command switches the console terminal from the MP Main Menu to mirrored/redirected console mode. All mirrored data is displayed. Type Ctrl/B to return to the MP command interface.

For VT100 terminals, verify that the MP setting in the CA command is correct and all mirrored consoles are of the same terminal type for proper operation.

Connect to Service Processor

CSP: Connect to remote management processor over the LAN.

This command allows the local or remote port user to connect over the MP LAN to another MP on the network. The user that launches the command is given a private connection to the other MP over the LAN. To return to the original MP, type CTRL-] to disconnect the CSP session.

Date

DATE: Displays the current date, as generated in the MP real-time clock.

Default Configuration

DC: Default Configuration—reset all MP parameters to the default configuration.

This command sets all MP parameters back to their default values. The user may reset all or a subset of the following parameters:

- IP configurations
- Modem configuration
- Paging configuration
- Command Interface configuration
- Disable remote access, security configuration
- Session configuration. For example, setting the security configuration to default erases all users.

There are three ways to reset passwords in the MP:

1. In the SO command, change individual users.
2. In the DC command, choose “Reset Security Configuration”.
3. Pressing the reset button on the back panel of your HP server can reset forgotten passwords. After the MP reboots, the local console terminal displays a message for five seconds. Responding to this message in time allows a local user to reset the passwords.

Pressing the reset button on the back panel of your HP Server can reset forgotten passwords. After the MP reboots, the local console terminal displays a message for five seconds. Responding to this message in time allows a local user to reset the passwords.

Notice that all user information (logins, passwords, and so on) is erased in methods 2 and 3.

Display FRUID

This command displays FRUID information from the BMC for FRU devices. Information provided includes serial number, part number, model designation, name and version number, and manufacturer.

Disconnect Remote or LAN Console

This command disconnects (hangs up) the remote/modem or LAN/WEB users from MP. It does not disable the ports. The remote console is no longer mirrored.

Front Panel Process

FP: Turn off front panel fault or attention LEDs.

This command allows the user to control the state of front panel fault and attention LEDs, individually or together.

MP Firmware Update

FW: Activates MP firmware upgrade mode.

This command is available from either the LAN or local serial port. This command activates firmware upgrade mode, which loads new firmware through the MP LAN by FTP (which must be operational). An MP Reset is generated after the upgrade is complete.

Help

HE: Display help for menu or command

This command displays the MP hardware and firmware version identity and the date and time of firmware generation. If executed from the MP Main Menu, general information about the MP and those commands displayed in the MP Main Menu are displayed. If executed in command mode, displays a list of command interface commands available to the user. Displays detailed help information in response to a topic or command at the help prompt.

Display System ID

ID: Display/modify system information.

This command allows the user to display and modify the following:

- SNMP contact information
- SNMP server information
- SPU hostname

Inactivity TImeout

IT: Inactivity Timeout settings

The session inactivity timeout is up to 1,440 minutes. The default is 60 minutes. This timeout prevents sessions to the system from being inadvertently left open. A session can be started by the SE command. An open session can prevent users from logging onto the MP through a port and can also prevent system applications from initiating an outbound connection.

MP inactivity timeout is up to 1,440 minutes. The default is 5 minutes. This timeout prevents a user from inadvertently keeping the MP locked in MP Command Interface mode preventing other users from looking at the console output. The MP Command Interface inactivity timeout may not be deactivated.

Flow control timeout is 0 to 60 minutes. If it is set to 0, no timeout is applied. This timeout prevents mirrored flow control from blocking other ports when inactive.

Configure LAN Console

LC: LAN configuration (IP address, and so on)

This command displays and allows modification of the LAN configuration. Configurable parameters include:

- MP IP Address
- MP Host Name
- Subnet Mask

- Gateway Address
- Web Console port number
- Link State

The MP Host Name set in this command is displayed at the MP command interface prompt. Typically the DNS name for the LAN IP is entered.

This field can be programmed to any useful name or phrase. For clarity, it is useful to enter MP-on-STST~4 as the MP Host name, so both names show up in the prompt (limit 19 characters, no spaces allowed.) The web access port number is also set by this command.

LAN Status

This command displays all parameters and the current status of the MP LAN connections. The LAN parameters are not modified by the execution of this command.

Return to Main Menu

MA: Return to MP Main Menu

This command makes the MP return to the non-mirrored MP Main Menu. This is the same as executing Ctrl/B.

Modem Reset

MR: Modem Reset

This command makes the MP send an AT Z command to the modem, which resets it. Any modem connections are lost. The initialization results can be viewed by using the MS command.

Modem Status

MS: Modem Status—Display modem status

The MS command displays the state of the modem lines connected to the remote/modem serial port. The display can be updated by pressing Enter. The current state of the status signals DCD, CTS, DSR, RI and the last state of the control signals DTR, RTS set by the firmware are displayed.

Power Control

PC: Power Control—turn system power on and off.

For proper system shutdown, shutdown the operating system before issuing this command or use the commands graceful shutdown option.

This command allows you to switch the system power on or off. You can have the action take place immediately or after a specified delay.

Notice this is roughly the equivalent to turning the system power off at the front panel switch. There is no signal sent to the OS to bring the software down before power is turned off. To turn the system off properly, you must ensure that the OS is in the proper shutdown state before issuing this command. Use the proper OS commands or use the graceful shutdown option of the Remote Power Control command.

Configure Paging

PG: Paging parameter setup—configures pagers.

This command allows the user to configure the pagers and set triggering events. A string description of the triggering event will be sent with the page.

Power Status

PS: Power status—display the status of the power management module.

This command displays on the console the status of the power management module.

Reset BMC

RB: Reset BMC.

This command resets the BMC by toggling a GPIO pin.

Reset System

RS: Reset system through RST signal.

This command causes the system (except the MP) to be reset through the RST signal.

Execution of this command irrecoverably halts all system processing and I/O activity and restarts the computer system. The effect of this command is very similar to cycling the system power. The OS is not notified, no dump is taken on the way down, and so on.

Set Access

SA: Set access options—configures access for LAN and remote/modem ports

This command will disconnect modem, LAN, and web users if access is disabled.

Create Local Session

SE: Log into the system on local or remote port

Only valid from the local or remote/modem port, SE allows the user to leave the MP Command Interface and enter a system session. Other mirrored MP users are placed in console mode. The session user returns to the mirrored MP session on exit.

The MP regularly checks the activity of the session, closes the connection with the system, and, if the timeout period has elapsed, returns the port to mirroring. The timeout period is set with the IT command. On HP-UX, the SE command works on the local and remote ports.

Display Logs

SL: Display the contents of the system status logs

This command displays the content of the event logs that have been stored in non-volatile memory.

- System Event Log (SEL) - High attention events and errors
- Forward prgress - All events
- Current boot log - All events between “start of boot” and “boot complete’
- Previous boot log - The events from the previous boot
- The table below defines alert (or severity) levels.

Reading the system event log turns off the attention LED. Accessing this log is the only way to turn off the attention LED when it is flashing and alerts have not been acknowledged at the alert display level.

Events are encoded data that provide system information to the user. Some well-known names for similar data would be Chassis Codes or Post Codes. Intelligent hardware modules, the OS, and system firmware produce events. Use VFP to view the live events. Use SL to view log events.

Navigate within the logs as follows:

- Plus sign (+) – View the next block (forward in time).
- Minus sign (-) – View the previous block (backward in time).
- Carriage-return – View the next block in the previously selected direction (forward or backward in time).
- D – Dump the entire log for capture or analysis.
- F – First Entry.
- L – Last Entry.
- J – Jump to entry number.
- V – View mode configuration (text, keyword, hex).
- Question mark (?) – Display help menu.
- Q – Quit

Table D-1 Severity Levels

Severity	Definition
0	Minor forward progress
1	Major forward progress
2	Informational
3	Warning
5	Critical
7	Fatal

Security Options

SO: Configure security options and access control (users, passwords, and so on.).

This command modifies the security parameters of the MP, which include login timeouts and allowed password faults.

If configured, when you access the MP using the modem port, the MP hangs up and dials the user back. This does not work if Modem Presence is set to not always connected with the CA command.

If the mode is Single, the State is changed to disabled after the first login. A disabled user's login is not accepted.

Firmware Revision Status

Firmware Revision Status

SYSREV: Displays the revision status of firmware in the system processors.This command displays the revision status of firmware in the system processors.

System Status

SS: Displays the status of the system processors

The SS command displays the status of the system processors and which processor is the monarch.

Tell

TE: TEll—sends a message to other terminals.

Up to 80 characters can be typed in. The message is broadcast to the other mirrored clients. Users in a session or CSP are not shown the message.

Transfer of Control

TC: System reset through INIT or TOC (Transfer of Control) signal

Under normal operation, shut down the operating system before issuing this command.

This command causes the system to be reset through the INIT (or TOC) signal. Execution of this command irrecoverably halts all system processing and I/O activity and restarts the computer system. It is different from the RS command in that the processors are signaled to dump state on the way down.

User Configuration

UC: User Configuration—controls user access

This command allows an Administrator to add, modify, re-enable, or delete user logins. The Administrator can also enable or disable security warnings and change passwords.

Virtual Front Panel

VFP: Display Virtual Front Panel

When invoked, this command displays a current summary of system status, including the state of front panel LEDs.

There are two ways that the live display of events can be started:

1. Live Mode: Invoked from the VFP command at the MP prompt. To exit, type Q to quit the console.
2. Early Boot Mode: When the boot sequence for the system begins, the live VFP is invoked automatically. When boot finishes, you are automatically switched to console mode.

The LED state reflects the state of the front panel LEDs. When system power is off, the remote LED shows “off” even thought remote access may be enabled with the EL or ER commands.

Who

WHO: Displays a list of MP connected users.

This command displays the login name and operating mode (Main Menu, command, etc.) of the connected console client users and the port on which they are connected. For the LAN and WEB console clients the remote IP address is also displayed.

If the local console client user did not originate the MP command Interface session, there is always one default user listed for the local serial port: local user i. If the local console operator typed Ctrl/B, then the login name that the local operator is displayed instead.

Exit from MP

X: Exit from MP command interface and disconnect from the system.

This command disconnects the executing user from the system. This command is available from the local port.

Diagnostics

XD: Diagnostics and/or Reset of MP

This command allows the user to perform some simple checks to confirm the MP's health and its connectivity status. The following tests are available:

- MP Parameter Checksum
- Verify I2C conenction (get BMC Device ID)
- LAN connectivity test using ping
- Modem self-tests

Also, the MP can be reset from this command. An MP reset can be safely performed without affecting the operation of the server.

Avoiding Command Confirmation

Most of the commands will prompt you for confirmation before performing the command. You can bypass this confirmation by adding "-nc" to the command line.

Gaining Control of the Console

If you have multiple users sharing a console, you will see the following message if you try to enter commands to the console:

```
Shell>

- - - - - - - - - - - - Live Console - - - - - - - - - - - -

[Read only - use Ctrl-Ecf for console write access.]

To get control you need to enter <ESC>cf, not <CTRL-E>cf.
```

E BMC Console Commands

This appendix describes the BMC system console and provides a summary of its commands.

BMC System Console

On the rx2600, the Baseboard Management Console (BMC) is accessible through the serial port on the back of the system labeled Console/Serial A. This port is set to 9600 baud 8-bit no parity and one stop bit.

Entering BMC Console Command Mode

To get the console processors attention and enter command mode, enter <ESC> (. You receive the cli> prompt, at which point you can enter commands.

Exiting BMC Console Command Mode

To exit the console processor command mode, enter <ESC>Q. If the OpenVMS operating system is running, you are returned to the operating system prompt.

Table E-1 lists the BMC commands.

Table E-1 BMC Command Summary

Command	Description
C[<passwordstring>]	Change password
CON	Display console selection
FPL	Read forward progress log
H	Help
I<ipmi command data>	Send any IPMI message
	req fmt: rsSA netfn1un chk1 rsSWID rqSeq cmd [data] chk2
	resp fmt: rqSWID netfn1un chk3 rsSA rqSeqrsLun cmd ccode [data] chk4
INFO	Display BMC FW revision
IPMI <ipmi command data>	Send any IPMI message
	req fmt: netfn1un cmd [data]
	resp fmt: ccode [data]
LOC[0,1]	Locator LED control
P[0,1]	Power control

Table E-1 BMC Command Summary (Continued)

Command	Description
Q	Quit/Logout
RS [s]	Reset system [and switch to sys console]
SD	Read SDR repository
SE	Read System Event Log
TOC[s]	Send an INIT/TOC [and switch to sys console]

F Alpha Tools

This appendix describes the Alpha tools that ship with the HP OpenVMS Industry Standard 64 Integrity Server Version 8.1. These tools allow you to perform tasks on your Alpha system that help you set up and manage your Integrity Server.

The following tools are included:

- INITIALIZE
- BACKUP
- SETBOOT
- SDA

The Alpha tools reside on the layered product media (DVD 2) in the [IA64_TOOLS]DIRECTORY. You install the tools through a PCSI procedure:

```
$ PRODUCT INSTALL I64_TOOLS
```

To use the tools, you must run two command procedures, which are copied to your system by the installation process:

```
$ @SYS$COMMON:[SYS$STARTUP]I64_TOOLS$STARTUP.COM
```

```
$ @SYS$COMMON:[SYSMGR]I64_TOOLS$SYLOGIN.COM
```

The startup procedure defines some logical names that allow you to use the Alpha tools, while not affecting normal operation of the Alpha system.

The sylogin procedure is run by a user who wants to use the tools in the I64_tools kit. Do not add this procedure to the default system login, as the procedure affects normal operation of several utilities on your Alpha system (INIT, BACKUP, SET, and SDA). The default SYSLOGIN.COM procedure maintains the normal Alpha operating environment systemwide for all users. Only the user intending to use the Alpha tools should run the I64_TOOLS$SYLOGIN.COM procedure.

Running the I64_TOOLS$SYLOGIN.COM procedure does the following:

- Modifies the INITIALIZE, BACKUP, and SET commands.
- Defines logical names to cause those DCL commands to execute the I64_TOOLS version of those images.
- Defines a few DCL global symbols.

If you specify 1 as the P1 when you execute the I64_TOOLS$SYLOGIN.COM procedure, you can see the logical names and DCL global symbols that are defined in the screen output.

The following sections provide more detail on the tools provided.

INITIALIZE

A new /GPT qualifier has been added to the INITIALIZE command.

$ INITIALIZE/[NO]GPT ddcu: volume-label

This new qualifier allows the user to specify the GUID Partition Table (GPT) when initializing a disk device. The default for INITIALIZE on I64 is /GPT.

BACKUP

The BACKUP image provided in the Alpha tools kit knows how to handle the I64 boot block setup.

The BACKUP utility has two new warning messages:

- BACKUP-W-GETBOOERR, error reading boot information
- BACKUP-W-SETBOOTERR, error writing boot information - volume not bootable

SDA

This tool allows you to view an I64 system crash dump on your Alpha system.

SET BOOTBLOCK

This command provides a raw interface to the DCL SET BOOTBLOCK command, so that you can create an I64 bootable device on an Alpha system. A jacket procedure checks to make sure that the MOVEFILE file attribute is set and that the EFI image is contiguous.

Currently, support exists for writing I64 boot blocks.

This DCL verb initializes the bootblock for a specified system disk device.

A description of the SET BOOTBLOCK command follows.

SET BOOTBLOCK

The SET BOOTBLOCK command initializes the bootblock on the target device.

Format

SET BOOTBLOCK [bootfile]

Parameter

bootfile

Specifies the filename of the bootfile for the target disk.

Architecture-specific defaults are applied for the bootfile:

I64: file SYS$EFI, default SYS$SYSROOT:[SYS$LDR]SYS$EFI.SYS

The bootfile must be contiguous. If the target bootfile is not contiguous, use COPY/CONTIGUOUS or another functionally similar mechanism to recreate a contiguous version of the bootfile.

The bootfile must also be marked NOMOVE (SET FILE/NOMOVE) to avoid bootstrap failure that could otherwise arise from the normal and expected operations of disk defragmentation tools.

Description

The SET BOOTBLOCK command writes a bootblock onto the specified disk.

SET BOOTBLOCK is a DCL-level command. However, you can also invoke the SETBOOT utility using RUN, and you are prompted for all required input.

NOTE SET BOOTBLOCK can generate OpenVMS I64 bootblocks.

Qualifiers

/BLOCK_SIZE=size

Specifies the target block size for the bootstrap device, in bytes. The 512 byte block size is the default and is used for most disk devices, while the 2048 byte block size is typically used only with specific OpenVMS I64 CD and DVD devices.

This value applies only to writing the boot block onto specific CD and DVD disk devices, as OpenVMS itself and most applications will assume 512 byte blocks for all disks.

Permitted values for the target block size are 512 and 2048.

This qualifier is valid only in conjunction with /I64.

/I64

Specifies OpenVMS I64 as the target architecture for the boot block. The default bootfile for OpenVMS I64 is SYS$SYSROOT:[SYS$LDR]SYS$EFI.SYS.

```
$ SET BOOTBLOCK/I64
```

/PRESERVE=keyword

/PRESERVE=SIGNATURE maintains the existing GUID disk signature value. The default is to generate a new signature for the target volume.

No other keywords are presently documented.

Permissible only with /I64.