1

OpenVMS I/O Version 7.1 New Features

1.1 High SCSI ID's on Wide Adapters (Alpha)

New feature for V7.1-FT1—never previously published

OpenVMS Alpha Version 7.1 wide adapter support of high SCSI IDs allows devices that support the wide SCSI bus option to use existing hardware more fully for device identification.

For adapters that support a wide SCSI bus option, high SCSI IDs enable configuration and use of up to 32 devices per SCSI bus. KZTSA, KZPSA, and the QLogic ISP1020 wide adapters currently allow use of 16 or 32 data lines for device identification.

Note that although the maximum allowable number of devices per SCSI bus has changed, the SCSI bus length requirements have not. When configuring devices above the previous limit of 8 devices, failure to adhere to bus length requirements can produce unpredictable results.

Supported SCSI bus lengths are as follows:

- 3 meters (10 feet) on fast buses
- 6 meters (20 feet) on slow buses
- Narrow devices do not talk to hosts above ID 7

To configure wide IDs on a BA356 box, refer to the BA356 manual *StorageWorks Solutions BA356-SB 16-Bit Shelf User's Guide* (Order Number EK-BA356-UG).

When a wide device is configured on a narrow bus, the typically device reports itself in the lower 8 IDs appropriate to the slot in the box where it resides. For instance, in the following example of wide and narrow disks in a BA356 box, disks 800, 900, 1000 and 1200 are wide disks. Disk 300 is a narrow device.

```
Slot 0 $1$DKA800:
Slot 1 $1$DKA900:
Slot 2 $1$DKA1000:
Slot 3 $1$DKA300:
Slot 4 $1$DKA1200:
```

SCSI IDs above 700 are displayed as `Invalid SCSI IDs` by an error logging utility.

1.2 OpenVMS SCSI Device Driver Enhancements

New info for V7.1-FT1—never previously published. Should this be included in the release notes instead of new features?????

OpenVMS SCSI device drivers have been enhanced significantly for OpenVMS Version 7.1. These enhancements allow OpenVMS SCSI device drivers to work with a wider range of non-Digital SCSI devices, while taking advantage of as much SCSI functionality as the devices offer. In most cases, the OpenVMS Version 7.1 SCSI driver enhancements eliminate the errors that occurred when

some non-Digital devices were mounted after booting Version 6.2 of the OpenVMS operating system.

1.3 File-based Autoconfiguration for Device Drivers

Note for V7.1-FT1 Reviewers

New feature for V7.1-FT1—never previously published.

I am still trying to understand this info and figure out the best way to document it for customers. I will send out what I come up with before Wednesday in a separate mail message. I don't want to delay distributing this draft any longer.

-Margie

1.4 Port Allocation Classes for Naming SCSI Devices in an OpenVMS Cluster

Note for V7.1-FT1 Reviewers

This new SCSI device naming information was written by and is the responsibility of the vacationing Suzy Kane. I will include any "must be fixed for FT1" comments; Suzy will handle any additional comments or issues about this topic after she returns on June 24, 1996.

-Margie

Port allocation classes are a new feature for naming SCSI devices attached to Alpha systems in an OpenVMS Cluster. The new device names are also supported on OpenVMS VAX via the MSCP server.

Port allocation classes are designed to solve the naming and configuration conflicts that can occur in a large OpenVMS Cluster that includes multiple hosts sharing one or more SCSI buses and multiple disks, some connected to the shared SCSI interconnect and some connected to single-host SCSI interconnects.

1.4.1 Overview of OpenVMS Cluster Device Naming

To understand the reasons for the creation of port allocation classes, it is important to understand how device names are derived for use in an OpenVMS Cluster system and the SCSI device naming requirements for a multi-host interconnect.

In the OpenVMS operating system, a device name takes the form of ddcu where:

- *dd* represents the predefined code for the device type,
- *c* represents the predefined controller designation
- *u* represents the unit number

In an OpenVMS Cluster environment, if a device is attached to a single computer or HSC subsystem, the name is extended to include a node name:

`node$ddcu`

where *node* represents the node name of the system on which the device resides

In an OpenVMS Cluster environment, where computers or HSC subsystems are dual pathed, the computers (or HSC subsystems) and every device to which there is a dual path must be given an allocation class. The allocation class, prefaced with a dollar sign ($), is added to the beginning of the name and the node name is dropped, as shown next:

```
$allocationclass$ddcu
```

1.4.2 Device Naming in a Cluster With a Shared SCSI Bus

In addition to the naming conventions described for devices in an OpenVMS Cluster where computers or HSC subsystems are dual pathed, additional similar naming requirements that affect device names are introduced in a cluster with multiple hosts on a shared SCSI bus. These requirements follow:

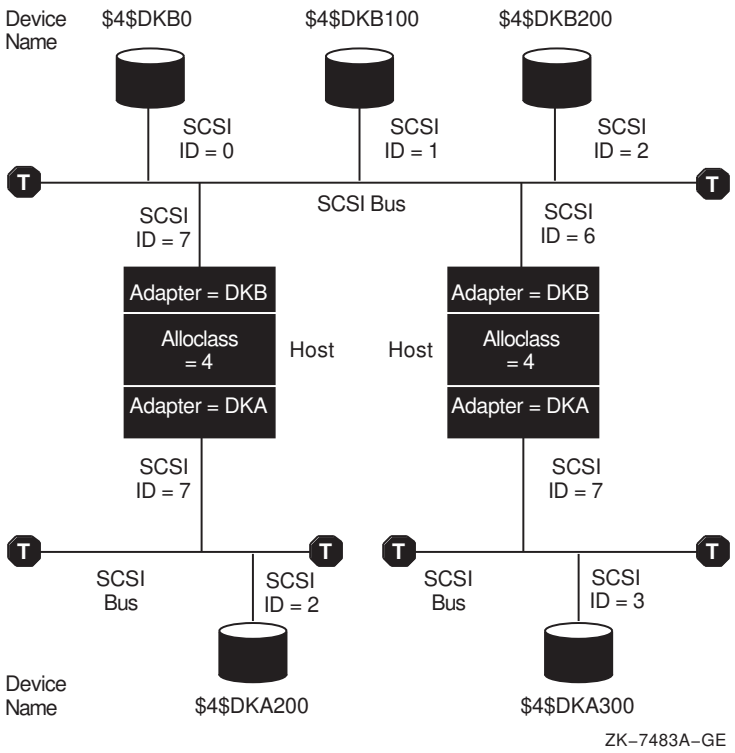
- All host controllers attached to a given SCSI interconnect must have the same OpenVMS device name, including the same controller letter, (for example, B in DKB*n*).
- Each system attached to a shared SCSI interconnect and each device attached to those systems (either to the shared SCSI interconnect or to a separate, private SCSI interconnect) must have the same nonzero allocation class.

  Consequently, the *ddcu* part of the disk device name (for example, DKB0), whether connected to a shared SCSI interconnect or to a single-host SCSI interconnect, must be unique within an allocation class.

A sample configuration of two hosts attached to a SCSI bus with shared disks is shown in Figure 1–1. Note the following:

- Each host also has a single-host (private) SCSI bus for additional storage.
- All disks use the same allocation class (the disks on the shared SCSI bus and the disks on the private SCSI buses).
- The two hosts cannot directly access each other's private disks, but those disks must still have unique names. If the controller letters are the same, the unit numbers must be different.

Figure 1–1 Device Names Prior to the Use of Port Allocation Classes



Additional naming and configuration constraints are imposed by the SCSI hardware, as described next:

- SCSI bus

  Usually only seven numbers (0 through 6) available on the bus to assign to devices and controllers

- Short length of SCSI cable, some of which may be used internally

  On some computers, part of the port A SCSI bus is used internally, leaving a relatively short length outside the computer. For the port B bus, the entire length of the SCSI bus is external.

Furthermore, if you want to configure small and large computer systems on the same SCSI bus, you may encounter another naming restriction. Typically, small computers have only one port named A or two ports named A and B. Large computers have multiple ports but ports A and B may not be available to connect to the SCSI bus. For example, on AlphaServer 8400 systems, ports A through D are used internally; ports starting with E are available for connecting to external buses.

Prior to the introduction of port allocation classes, the SCSI naming requirements and the SCSI hardware constraints created some situations where nodes sharing a SCSI bus had empty drive bays. Customers could not put disks in them because it would have caused two or more devices to have the same device name.

1.4.3 How Port Allocation Classes Work

A port allocation class is a designation for all ports attached to a single interconnect. It replaces the node allocation class in the device name. The use of port allocation classes removes the SCSI naming requirements described in Section 1.4.2.

There are three types of port allocation classes:

- Port allocation classes of 1 to 32767 for devices attached to a multi-host interconnect or a single-host interconnect, if desired
- Port allocation class 0 for devices attached to a single-host interconnect
- Port allocation class -1, which means no port allocation class is in effect

Each type has its own naming rules.

1.4.3.1 Port Allocation Classes for Devices Attached to a Multi-Host Interconnect

The following rules pertain to port allocation classes for devices attached to a multi-host interconnect:

1. The valid range of port allocation classes is 1 through 32767
2. When using port allocation classes, the controller letter in the device name is always A, regardless of the actual controller letter[1]
3. Port allocation classes must be unique within a cluster
4. A port allocation class cannot duplicate the value of another node's tape or disk allocation class

Examples of device names that use this type of port allocation class are shown in Table 1–1.

Table 1–1 Examples of Device Names With Port Allocation Classes 1-32767

Device Name	Description
101DKA0	The port allocation class is 101; DK represents the disk device category, A is the controller name requirement, and 0 is the unit number.
147DKA0	The port allocation class is 147; DK represents the disk device category, A is the controller name requirement, and 0 is the unit number.

The use of a port allocation class for a multi-host SCSI interconnect removes the following SCSI naming restrictions:

1. All devices attached to a host, which shares a SCSI interconnect, must use the same node allocation class, regardless of whether the devices can be accessed by the other hosts sharing the SCSI interconnect

   Port allocation classes replace node allocation classes.

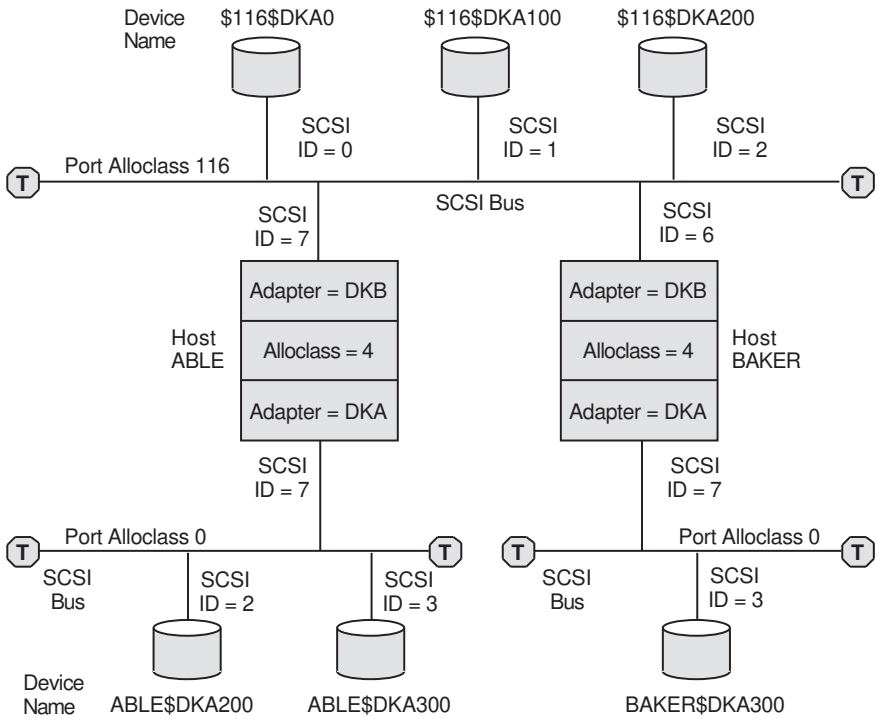
2. The controllers of all hosts attached to the interconnect must have the same designation.

   The controllers no longer must have the same designation. OpenVMS assigns the letter A to all controllers, regardless of the actual controller letter.

[1] The $GETDVI item code DVI$_DISPLAY_DEVNAM displays the actual port name.

Figure 1–2 shows the same configuration as Figure 1–1, except for an additional disk attached to the lower left SCSI bus. In Figure 1–2, port allocation classes are used in device names. A port allocation class of 116 is used for the SCSI interconnect that is shared, and port allocation class 0 is used for the SCSI interconnects that are not shared.

Figure 1–2 Device Names Using Port Allocation Classes



1.4.3.2 Port Allocation Class 0 for Devices Attached to a Single-Host Interconnect

The following rules pertain to port allocation class 0 for devices attached to a single-host interconnect:

1. Port allocation class 0 does not become part of the device name. Instead, the name of the node to which the device is attached becomes the first part of the device name.
2. The controller letter in the device name remains the designation of the controller to which the device is attached. (It is not changed to A as it is for port allocation classes greater than zero.)

Examples of device names that use port allocation class 0 are shown in Table 1–2.

Table 1–2 Examples of Device Names With Port Allocation Class 0

Device Name	Description
ABLE$DKD100	ABLE is the name of the node to which the device is attached. D is the designation of the controller to which it is attached, not A as it is for port allocation classes with a non-0 class. The unit number of this device is 100. The port allocation class of 0 is not included in the device name.
BAKER$DKC200	BAKER is the name of the node to which the device is attached, C is the designation of the controller to which it is attached, and 200 is the unit number. The port allocation class of 0 is not included in the device name.

The use of port allocation class 0 for a single-host SCSI interconnect removes the following SCSI naming restrictions:

1. All devices attached to a host, which shares a SCSI interconnect, must use the same node allocation class, regardless of whether the devices can be accessed by the other hosts sharing the SCSI interconnect
2. The *ddcu* part of the device name of all devices attached to a host that shares a SCSI interconnect must be unique

   Because the names of devices that use port allocation class 0 also use the node name as part of the device name, multiple disks with the same name, such as DKA300, can exist in a cluster with a shared SCSI interconnect. For example, ABLE$DKA300 and BAKER$DKA300 are valid names, as shown in Figure 1–2.

1.4.3.3 Port Allocation Class -1

The designation of port allocation class -1 means that a port allocation class is not being used. Instead, the allocation class based on the node remains in effect.

1.4.4 How to Implement Port Allocation Classes

To implement port allocation classes, you must do the following:

- Enable the use of port allocation classes
- Assign one or more port allocation classes
- At a minimum, reboot the nodes on the shared SCSI bus

1.4.4.1 Enabling the Use of Port Allocation Classes

To enable the use of port allocation classes, you must set a new SYSGEN parameter DEVICE_NAMING to 1. The default setting for this parameter is 0. In addition, the SCSSYSTEMIDH system parameter must be set to 0. Check to make sure that it is.

1.4.4.2 Assigning Port Allocation Classes

You can assign one or more port allocation classes with the OpenVMS Cluster configuration procedure, CLUSTER_CONFIG.COM (or CLUSTER_CONFIG_LAN.COM).

If it is not possible to use CLUSTER_CONFIG.COM or CLUSTER_CONFIG_LAN.COM to assign port allocation classes, for example, if you are booting a private system disk into an existing cluster, you can use the new SYSBOOT SET/CLASS command.

The following example shows how to use the new SYSBOOT SET/CLASS command to assign an existing port allocation class of 152 to port PKB.

```
SYSBOOT> SET/CLASS PKB 152
```

The SYSINIT process will ensure that this new name is used in successive boots.

To deassign a port allocation class, enter the port name without a class number; for example:

```
SYSBOOT> SET/CLASS PKB
```

The mapping of ports to allocation classes is stored in SYS$SYSROOT:SYS$DEVICES.DAT, a standard text file. You use the CLUSTER_CONFIG.COM (or CLUSTER_CONFIG_LAN.COM) command procedure or, in special cases, SYSBOOT, to change SYS$DEVICES.DAT.

1.4.4.3 Cluster-Wide Reboot Requirements

Changing a device's allocation class changes the device name. A cluster-wide reboot ensures that all nodes see the device under its new name, which in turn means that the normal device and file locks remain consistent.

Rebooting an entire cluster when a device name changes is not mandatory. It may be possible to reboot only the nodes that share the SCSI bus, as described in the following steps. The conditions under which you can do this and the results that follow are also described.

1. Dismount the devices whose names have changed from all nodes

   This is not always possible. In particular, you cannot dismount a disk on nodes where it is the system disk. If the disk is not dismounted, a subsequent attempt to mount the same disk using the new device name will fail with the following error:

   ```
   %MOUNT-F-VOLALRMNT, another volume of same label already mounted
   ```

   Therefore, you must reboot any node which cannot dismount the disk.

2. Reboot all nodes connected to the SCSI bus

   Before you reboot any of these nodes, make sure the disks on the SCSI bus are dismounted on the nodes not rebooting.

   Note

   OpenVMS ensures that a node cannot boot if the result is a SCSI bus with naming different from another node already accessing the same bus. (This check is independent of the dismount check in the previous step.)

   After the nodes that are connected to the SCSI bus reboot, the device exists with its new name.

3. Mount the devices system- or cluster-wide.

   If no other node has the disk mounted under the old name, you can mount the disk system- or cluster-wide using its new name. The new device name will be seen on all nodes running compatible software, and these nodes can also mount the disk and access it normally.

Nodes that have not rebooted will still see the old device name as well as the new device name. However, the old device name cannot be used; the device, when accessed by the old name, is off line. The old name persists until the node reboots.

2

OpenVMS I/O Version 7.1 Release Notes

2.1 Device Support on OpenVMS Alpha Systems

Alpha

The following sections contain release notes pertaining to OpenVMS Alpha device drivers.

2.1.1 Changes and Enhancements

This section describes OpenVMS Alpha device support changes or enhancements.

2.1.1.1 OpenVMS Alpha and OpenVMS VAX SCSI Device Drivers Must Be Recompiled and Relinked for Version 7.1

New note for V7.1-FT1—never previously published.

OpenVMS Version 7.1 includes changes to the data structures used by OpenVMS Alpha and OpenVMS VAX SCSI device drivers.

As a result of these changes, all OpenVMS Alpha and OpenVMSVAX SCSI device drivers from previous versions of OpenVMS must be recompiled and relinked to run correctly on OpenVMS Version 7.1.

2.1.1.2 Using the SCSI_MODE Utility to Disable Tagged Command Queuing

New note for V7.1-FT1—never previously published. Rick Lord and I are still working on this one.

TBS

2.1.1.3 Configuring an ISA device in an EISA Slot

New note for V7.1-FT1—never previously published. Although, I included this in the new Writing Drivers in C book, which will be available in a few weeks.

Because ISA devices do not have a software-readable ID, they cannot be detected by operating system software or by console firmware. Therefore, no information is displayed for EISA slots populated by ISA devices.

To create an ECU entry for an ISA device, a configuration file for the device must exist either on the ECU diskette or on another FAT-format diskette.

The ECU provided by Digital contains a generic ISA configuration file, ISA0000.CFG, that can serve as an example. This configuration file can be copied to another diskette and edited using a personal computer. The EISA specification provides details on configuration file syntax and format.

To create the ECU entry for an ISA device in an EISA slot:

1. Run the ECU from the console prompt.
2. From the ECU main menu, choose Step 2 (Add or remove board).

3. Select a slot for the ISA card and press enter.
4. Insert the diskette containing the configuration file for the ISA device and press enter.
5. Press F10 to exit to the main menu.
6. Choose Step 3 (View or edit details).
7. Examine the information for the slot and adjust as needed. To change a resource, position the cursor to that resource and press F6.
8. Press F10 to exit to the main menu.
9. Select Step 5 and exit, saving the changes.

After creating the ECU entry, adjust any necessary jumpers on the card to reflect the assigned IRQ, DMA channels, etc. Insert the card in the chosen slot.

To connect an ISA device in an EISA slot, execute the ECU as described in the previous procedure. Make a note of the slot number, because this information will be required for the /NODE input to SYSMAN. Compute the system_IRQ as described for EISA devices. Use the TR of the EISA bus for the /ADAPTER parameter.

All ISA devices use the same CSR. By convention, ISA I/O address space is accessed as EISA slot 0 (ISA devices ignore the 4-bit slot portion of the EISA I/O space address). Therefore, the CSR for all ISA devices is the base virtual address of EISA slot 0, the base of EISA I/O space. To obtain this value, use the SDA command CLUE CONFIG to locate the EISA ADP data structure. Format the EISA ADP and use the value in the `adp$q_csr` as the CSR parameter for all ISA cards.

2.1.1.3.1 Example: Locating the CSR for an ISA Device in an EISA Slot

```
$ ana/sys
SDA> read sys$loadable_images:iodef
SDA> clue config
Adapter Configuration:
----------------------
TR  Adapter Name (Address)  Hose  Bus          Node  Device Name       HW-Id/SW
--  ----------------------  ----  -----------  ----  ----------------  --------
 1              (81133480)     0  CBUS
                                                0  KA0902_CPU        00000017
                                                5  KA0902_MEM        00000018
                                                8  KA0902_IIO        00000019
 2  PCI         (81133880)     0  PCI
                                                2  MERCURY           04828086
                                        GYA:    7  TGA               00041011
 3  EISA        (81133F00)     0  EISA
                                                0                    012AA310
                                        PAA:    2  KFESA            002EA310
                                        PAB:    3  KFESA            002EA310
 4  XBUS        (81134440)     0  XBUS

SDA> ex 81133f00 + adp$q_csr          ! The EISA ADP CSR is the base for slot 0
81133F00:  FFFFFFFF 87AE0000          ! All ISA devices use CSR 87AE0000
```

2.1.1.4 Naming Serial Line Devices

New note for V7.1-FT1—never previously published.

OpenVMS has changed the way it handles the second serial port on the following Alpha systems:

Alphastation 200,400, 500 and 600 families
Alphaserver 1000,2000 and 2100 families

If one of these systems is booted with the console environment variable set to graphics, the name of the serial line (COM1) will be different than in previous releases. The COM1 device is called TTB0, instead of OPA1. In this case, the COM1 device is controlled by SYS$YSDRIVER, instead of SYS$OPDRIVER.

If the console is set to serial, the COM1 device is called OPA0.

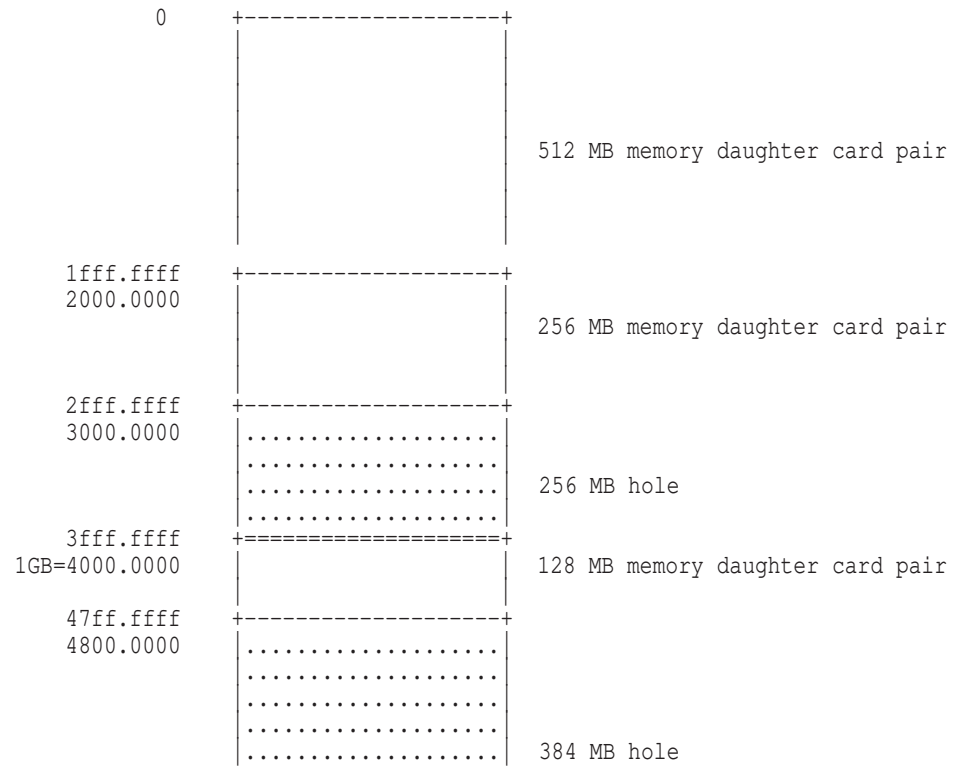
2.1.1.5 Memory Holes

New note for V7.1-FT1—never previously published. Walt and I know this needs work, and we're working on it. Here's what we have so far. I'll replace "Rawhide" with the official name. I'm having "real" art drawn, too.

As illustrated in the diagram in this section, memory holes might exist on Rawhide. In this case there are three different sizes of memory daughter card pairs: 512 MB, 256 MB and 128 MB. By configuration rules these muse be arranged in descending order.

The hardware reads the first set of memory daughter cards and assumes that any following are the same size. Since this is not the case, memory holes occur. The one at 3000.0000 must be dealt with by OpenVMS. The one at 4800.0000 is at the top of the address space, and can be ignored by OpenVMS.

Previous versions of OpenVMS did not efficiently support systems with physical memory holes and ultimately led to an inefficient use of the system's memory. The memory management data structures in OpenVMS Alpha Version 7.1 have been slightly modified to recognize the holes such that the inefficiencies in previous versions have been eliminated.




```
mmg$gl_memsize = 20000 (regardless of setting of SET MEM)
mmg$gl_maxpfn  = 23fff (regardless of setting of SET MEM)
```

Note that this configuration impacts the alogithm used to determine whether a driver needs to use map registers. Today drivers do the following:

1. Call IOC$NODE_DATA with key IOC$K_DIRECT_DMA_SIZE to obtain the size of the direct DMA window (in megabytes). This is usually 1 GB.
2. Convert the size returned from IOC$NODE_DATA to pages
3. Compare with mmg$gl_memsize
4. If size is greater than mmg$gl_memsize, use map registers; otherwise, use the direct DMA window. Because mmg$gl_memsize does **not** include the hole, the system appears to have only 1 GB of memory, and it appears that the device can use the direct DMA window. However, there is 128 MB of memory beyond the 1 GB border. For this reason, all drivers that use this algorithm should substitute it with the following:
   - Call IOC$NODE_DATA with key IOC$K_DIRECT_DMA_SIZE to obtain the size of the direct DMA window (in megabytes). This is usually 1 GB.
   - Convert the size returned from IOC$NODE_DATA to pages
   - Compare with (mmg$gl_maxpfn + 1)
   - If size is greater than (mmg$gl_maxpfn + 1), use map registers, otherwise use the direct DMA window.

2.1.1.6 OpenVMS Alpha Device Drivers Must Be Recompiled and Relinked for Version 7.0

Published for V7.0, I'm planning to publish it for V7.1-FT1.

V7.0

OpenVMS Alpha Version 7.0 includes significant changes to OpenVMS Alpha privileged interfaces and data structures. As a result of these changes, all device drivers from previous versions of OpenVMS Alpha must be recompiled and relinked to run correctly on OpenVMS Alpha Version 7.0.

For more details about OpenVMS Alpha Version 7.0 changes that may require source changes to customer-written drivers, see the *OpenVMS Alpha Guide to Upgrading Privileged-Code Applications*.

Important Note for OpenVMS Version 7.1

OpenVMS Alpha Device drivers that were recompiled and relinked for OpenVMS ALpha Version 7.0 do not have to be recompiled and relinked to run on OpenVMS Alpha Version 7.1. However, OpenVMS Alpha and VAX SCSI device drivers from any version prior to OpenVMS Version 7.1 must be recompiled and relinked as described in Section 2.1.1.1.

2.1.1.7 Bus Array Entry Structure Change

Published for V7.0—OK to delete for V7.1?????

V7.0

The bus array entry structure has been expanded for OpenVMS Alpha Version 7.0. The new bus array entry is one quadword larger than the old one.

This change requires any code which walks the bus array (for example, ICBMs used to auto-configure device drivers) to recompile and relink to run on OpenVMS Alpha Version 7.0.

2.1.1.8 SYS$MSBDRIVER Removed from OpenVMS Distribution

Published for V7.0—OK to delete for V7.1?????

V7.0

The driver for the Microsoft Windows Sound System ISA sound card (MSB), SYS$MSBDRIVER, has been removed from the OpenVMS Distribution as of Version 7.0. The following files have been removed:

- SYS$LOADABLE_IMAGES:SYS$MSBDRIVER.EXE
- SYS$EXAMPLES:SOUND_SERVICES.C
- SYS$EXAMPLES:SOUND_SAMPLE.C
- SYS$EXAMPLES:SOUND_SAMPLE.SND
- SYS$LIBRARY:SYS$STARLET_C.TLB module MSB.H

An enhanced version of this driver, called MMOV$MSBDRIVER, is included in Multimedia Services Version 1.4 for OpenVMS Alpha. This layered product also includes support for video capture and playback, an enhanced version of DECsound, and other audio and video applications.

MMOV$MSBDRIVER provides the same $QIO programming interface as SYS$MSBDRIVER. Digital recommends that the WAVE Applications Programming Interface provided by Multimedia Services for OpenVMS be used instead because it is more flexible and is portable to other platforms. (Multimedia Services Version 1.4 for OpenVMS is described in SPD 64.24.00.)

See Section 2.1.2.1 for information about a problem related to SYS$MSBDRIVER.

2.1.1.9 Enhanced SCSI Interface to IO$_DIAGNOSE Function

Published for V7.0—OK to delete for V7.1?????

V7.0

The $QIO IO$_DIAGNOSE function has been enhanced to include 64-bit addressing support for the following SCSI class drivers: GKDRIVER, DKDRIVER, and MKDRIVER. For more information, see the *OpenVMS Alpha Guide to 64-Bit Addressing*.

2.1.1.10 AlphaStation 205/255 Note: PCI Configuration Restriction

Published for V6.2—1H3. Should I publish or delete this for V7.1????

The OpenVMS Alpha operating system does not support PCI option cards configured in PCI slot 0 on any AlphaStation 205 and 255 series systems.

PCI slot 0 is the lowest physical PCI option slot on AlphaStation 205 and 255 series systems. The interrupt signal for this slot is shared with the built-in Ethernet port. Because the OpenVMS Alpha operating system does not currently permit PCI devices to share an interrupt line, a PCI device installed in slot 0 will not function correctly or might cause errors to occur with the built-in Ethernet port. As a result of this restriction, AlphaStation 205 and 255 series systems support a maximum of two PCI option cards, configured in slot 1 and slot 2.

2.1.1.11 PCI DMA Window Changes

Note for V7.1-FT1 Reviewers

This note was published for V7.0. The information is now included in the new Writing OpenVMS Alpha Device Drivers in C book, which will be available in time for FT2. I plan to publish this note for FT1 and delete it in the FT2 and SSB versions of the the release notes.

Comments or questions?????

V6.2

This release note should be read by anyone writing a driver for a PCI, EISA, or ISA device.

PCI bus implementations on Alpha platforms define two ways for PCI devices to access main memory: scatter/gather memory access and physical memory access. In **scatter/gather memory access**, the PCI address generated by a PCI device is translated to a main memory address by a scatter/gather table. In **physical memory access**, the PCI address generated by a PCI device is translated to a main memory address by the addition of a constant. Scatter/gather memory access is called **scatter/gather DMA**. Physical memory access is called **physical (or direct) DMA**.

Each memory access technique has advantages and disadvantages. Scatter/gather DMA allows access to all system memory, but it is more complex to program. Physical DMA is easier to program, but it may limit the amount of main memory that can be addressed.

Alpha PCI platforms implement scatter/gather DMA and physical DMA through **DMA address windows** on the PCI bus. A PCI DMA address window is an address range on the PCI bus through which PCI devices (and EISA/ISA devices behind the PCI/EISA bridge or PCI/ISA bridge) access main memory. The Alpha platforms that support PCI buses (AlphaServer 2000, AlphaServer 2100, AlphaServer 1000, AlphaServer 400, AlphaServer 8200, AlphaServer 8400, AlphaStation 200, AlphaStation 250) provide a minimum of two DMA address windows on the PCI bus. A DMA address window can be a physical DMA window, where a PCI bus address is a linear function of a system memory address, or a scatter/gather DMA window, where a PCI bus address undergoes a page table translation before becoming a main memory address.

Alpha platforms that support PCI buses have both a physical DMA window and a scatter/gather DMA window. In OpenVMS Alpha Version 6.2, the scatter/gather DMA window is based at PCI address 0 and extends to a maximum address of 3FFFFFFF (the actual size of the scatter/gather DMA window is a function of the amount of physical memory in the system). In OpenVMS Alpha Version 6.2, the physical DMA window is typically based at a PCI address above the scatter /gather DMA window.

To allow drivers to find the base of the physical DMA window in a platform-independent manner, two new function codes have been added to the bus support routine IOC$NODE_DATA. The basic definition of the IOC$NODE_DATA routine is as follows:

```
int ioc$node_data (CRB  *crb, int  function_code, void *user_buffer)
```

Inputs:

crb	Address of CRB. IOC$NODE_DATA uses the crb$l_node field and the vec$ps_adp field to find the data structures associated with the I/O bus to which this device is connected.
function_code	From [lib.lis]iocdef.sdl. Specifies information to be returned by IOC$NODE_DATA.
user_buffer	Address of caller's buffer. On success, the requested information is returned in the caller's buffer.

Outputs:

SS$_NORMAL	Normal successful completion. Requested information is returned in the caller's buffer.
SS$_ILLIOFUN	Unrecognized function code.
SS$_BADPARAM	Bad parameter. Usually this means that crb$l_node contains an invalid slot number. Check that the driver has been loaded with the /node qualifier.

When called with function code IOC$K_DIRECT_DMA_BASE, the IOC$NODE_ DATA routine returns the base address of the physical DMA window. The base address is returned as a 64-bit value in anticipation of future 64-bit I/O buses. For this reason, the caller of the IOC$NODE_DATA routine should make sure the user_buffer argument points to a quadword cell when using the IOC$K_DIRECT_ DMA_BASE function code.

When called with function code IOC$K_DIRECT_DMA_SIZE, the IOC$NODE_DATA routine returns the size of the physical DMA window, expressed in megabytes. The size of the direct DMA window is returned as a 32-bit value.

Using the physical DMA window for device DMA is straightforward. Once you have found the base of the physical DMA window using the IOC$NODE_DATA routine, you must adjust the DMA address that you assign to the device by adding the physical DMA window base to the main memory DMA buffer address. For example, on a typical system the physical DMA window is based at the PCI address 40000000 and extends to 7FFFFFFF. PCI addresses in the range from 40000000 to 7FFFFFFF are passed to main memory addresses 0 to 3FFFFFFF. This means that if you have a DMA buffer at main memory address 0, the PCI device would access this buffer at PCI address 40000000. The correspondence of main memory address and PCI DMA addresses is derived by the following formula:

```
PCI DMA address = (main memory buffer address) + (base of physical DMA window)
```

The main reason for locating the PCI DMA windows this way is to support bit-limited ISA devices. In OpenVMS Alpha Version 6.1, bit-limited ISA bus masters cannot use the scatter/gather DMA window because ISA bus masters do not have enough address bits to generate an address above 16 MB. For this reason, the scatter/gather DMA window was moved to PCI address 0. The scatter/gather table is now managed such that EISA and ISA devices that perform DMA in the scatter/gather window always get bus addresses below 16 MB.

Note that the size of the physical DMA window limits the amount of physical memory that can be addressed by a PCI, EISA, or ISA device. In the physical DMA window, it is not possible for an I/O device to address more than 1 GB of physical memory. For this reason, on large memory systems you may want to code your driver to perform DMA in the scatter/gather window. To perform scatter/gather DMA, use the standard counted resource management routines described in the Bookreader version of the *OpenVMS Alpha Device Support: Developer's Guide*.

2.1.1.12 Device IPL Setup for Drivers

Published for V6.2 and V7.0—OK to delete for V7.1?????

V6.2

Alpha hardware platforms that support PCI, EISA, and ISA buses deliver I/O device interrupts at different IPLs, either 20 or 21. The IPL at which device interrupts are delivered can change if you move the device from one platform to another. This is a problem if the driver declares its device IPL to be 20, and then that driver is executed on a machine that delivers I/O device interrupts at IPL 21.

The simplest solution to this problem is for PCI, EISA, and ISA device drivers to use IPL 21. This works correctly on platforms that deliver I/O device interrupts at IPL 20 and on platforms that deliver I/O device interrupts at IPL 21.

A future release of OpenVMS Alpha may provide a platform-independent mechanism for drivers to determine the device IPL dynamically.

2.1.1.13 SCSI Port and Class Driver Changes

Published for V6.2 and V7.0—OK to delete for V7.1?????

V6.2

The following sections describe changes to OpenVMS Alpha SCSI class and port device drivers.

2.1.1.13.1 OpenVMS Alpha SCSI Class Driver Changes

V6.2

Some SCSI-2 port interface (SPI) macros that supported OpenVMS Alpha SCSI class device drivers for Version 6.1 have been changed for OpenVMS Alpha Version 6.2. Table 2–1 lists the obsolete Version 6.1 macros and the Version 6.2 macros that have similar functions.

Note that many of the Version 6.2 SPI macros use different names, and most of them require new arguments. For this reason, do not replace each reference to an obsolete SPI macro without taking into account the OpenVMS Alpha Version 6.2 interface changes.

You can extract the new SPI$ interface macro definitions from SYS$LIBRARY:LIB.MLB, using the following command:

```
$ LIBRARY/EXTRACT=SPI$*/OUTPUT=SPI$DEFINITIONS.MAR SYS$LIBRARY:LIB.MLB
```

Table 2–1 Obsolete SPI Interface Macros

Obsolete Version 6.1 Macro	Similar Version 6.2 Macro
SPI$MAP_BUFFER	SPI$BUFFER_MAP
SPI$UNMAP_BUFFER	SPI$BUFFER_UNMAP
SPI$SEND_COMMAND	SPI$SEND_COMMAND
SPI$SET_CONNECTION_CHAR	SPI$CONNECTION_CHAR_SET
SPI$GET_CONNECTION_CHAR	SPI$CONNECTION_CHAR_GET
SPI$RESET	SPI$RESET_SCSI_BUS
SPI$ABORT_COMMAND	SPI$ABORT_COMMAND
SPI$DISCONNECT	SPI$DISCONNECT
SPI$ALLOCATE_COMMAND_BUFFER	SPI$CMD_BUFFER_ALLOC
SPI$DEALLOCATE_COMMAND_BUFFER	SPI$CMD_BUFFER_DEALLOC
SPI$RESERVED1	-
SPI$RESERVED2	-
SPI$CONNECT	SPI$CONNECT
SPI$FREEZE_QUEUE	SPI$QUEUE_FREEZE
SPI$TQE_WAIT	Use EXE$KP_TQE_WAIT
SPI$RELEASE_QUEUE	SPI$QUEUE_RELEASE
SPI$FLUSH_QUEUE	SPI$QUEUE_FLUSH
-	SPI$RESET_DEVICE

2.1.2 Problems and Restrictions

This section describes known problems and restrictions related to OpenVMS Alpha device support.

2.1.2.1 SYS$MSBDRIVER Load Failure

Published for V7.0—OK to delete for V7.1?????

V7.0

On systems with ISA-only I/O buses, OpenVMS Alpha Version 7.0 attempts to load SYS$MSBDRIVER under the following conditions:

- The sound card is configured at the console with the ISACFG console command.
- Multimedia Services for OpenVMS is not installed.
- The user issues the SYSMAN IO command AUTOCONFIGURE.

This attempt fails because SYS$MSBDRIVER is no longer available with OpenVMS (see Section 2.1.1.8). This error does not affect the operation of the system, except that the sound card is not usable.

If the SYSMAN IO command AUTOCONFIGURE is issued on these systems, the following error messages are displayed:

```
%IOGEN-E-LOADERR, error configuring device AUA0 driver SYS$MSBDRIVER
-RMS-F-FNF, file not found
```

To determine if the sound card is configured, use the ISACFG -ALL console command, and look for device name ''PCXBJ''. If this device is configured, you can use the ISACFG -RM command to remove the device from the configuration.

This problem will be corrected in a future release of the operating system.

2.1.2.2 SCSI Firmware Support

The following sections relate to SCSI firmware support.

2.1.2.2.1 Updating Firmware in SCSI Configurations

Published for V6.2—1H3 —I'm planning to publish for V7.1.

V6.2—1H3

When you update the firmware for a SCSI device, that device must be connected directly to a SCSI adapter on the host system. If you apply firmware updates on SCSI devices that are accessed through an intelligent controller (such as an HSZ40 or KZPSC), the firmware update will fail.

2.1.2.2.2 Required Firmware Support for RZ26N and RZ28M Disks

Published for V6.2—1H3 —I'm planning to publish for V7.1.

V6.2—1H3

The minimum firmware revision level required for RZ26N and RZ28M disks is Revision 0568.

If the latest firmware revision level is not used with these disks, multiple problems may occur.

2.1.2.2.3 SCSI Firmware Update Instructions

Published for V6.2 and V7.0—I'm planning to publish for V7.1.

V6.2

If your system contains RZ26L or RZ28 disks, the disks' firmware revision level must be at or above 442D for use in a SCSI OpenVMS Cluster system.

Important Note

Only certain RZ26L and RZ28 firmware revisions can be safely upgraded to firmware revision level 442D. Refer to Section 2.1.2.2.4 to determine if your disks are capable of being upgraded to firmware revision level 442D. If your disk is capable of supporting firmware revision level 442D, use the RZTOOLS Utility that is described in Section 2.1.2.2.5 to update the disk's firmware.

2.1.2.2.4 Firmware Revision Level 442D Requirements

Published for V6.2 and V7.0—I'm planning to publish for V7.1.

V6.2

Only the following combinations of disk drives and firmware revision levels are capable of being upgraded safely to firmware revision level 442D. Performing the update procedure on any other combination can permanently damage the disk.

Table 2–2 Revision Level 442D Firmware Compatibility

Disk Drive	Firmware Revision	Disk File Name
RZ26L	440C	RZ26L_442D_DEC.FUP
RZ28	441C or D41C	RZ28_442D_DEC2104.FUP

2.1.2.2.5 442D Firmware Installation Procedure

Published for V6.2 and V7.0—I'm planning to publish for V7.1.

V6.2

If you determine that your disk requires revision level 442D firmware, and it is capable of being upgraded safely, use the following procedure to update the firmware. (See Table 2–2 for the file name of the disk you are upgrading.)

```
$ MCR SYS$ETC:RZTOOLS_ALPHA DKB500 /LOAD=SYS$ETC:disk filename
  Read in 262144 bytes.
  Current FW version - X440C
  Upgrading to       - DEC0
  Loading code  ......
  New code has been sent to the drive.
```

2.1.2.3 SCSI Port and Class Drivers

V6.2

The following sections describe OpenVMS Alpha SCSI class and port device driver restrictions.

2.1.2.3.1 Mounting Third-Party SCSI Disks

This problem has been corrected. OK to delete for V7.1?????

V6.2

Users of the Digital RZ25M, Imperial MegaRAM MG-SCSI2, Seagate ST31200N, and ST31250N disks as well as other disks not qualified by Digital may see errors logged against these disks when they are mounted for the first time after booting in OpenVMS Alpha Version 6.2. Although these disks mounted error-free in OpenVMS Alpha Version 6.1, users might see 1 to 11 errors in OpenVMS Alpha Version 6.2 on the first mount.

The errors logged against these disks do not indicate a real problem. They occur because a 10-byte mode sense command is now issued to a disk instead of a 6-byte mode sense. This approach allows OpenVMS to work with a wider variety of third-party and other previously untried disks.

Digital is working with the parties responsible for maintaining the disk's firmware to correct this condition.

2.1.2.3.2 Add-On SCSI Adapters

Tom C says this has been enhanced by info in V6.2-1H3. I need to find the new info.

V6.2

OpenVMS Alpha Version 6.2 supports various add-on SCSI adapters. Digital's AlphaGeneration platforms typically support one or more integral SCSI adapters, with the option of installing additional add-on SCSI adapters. Due to differences in device-naming conventions used between the Alpha console and OpenVMS, the OpenVMS device name may not match the name displayed by the console.

For example, the console designation for a SCSI device on the integral SCSI adapter may be DKA100. However, when two additional add-on SCSI adapters are added, the ''A'' designation becomes ''C''; and DKA100 appears as DKC100 when OpenVMS is running.

Note that although the console and OpenVMS device names may be different, the unique specification of a device name from the console to the device name under OpenVMS will stay consistent, provided add-on SCSI adapters are not added or removed.

2.1.2.3.3 HSZ40 and Transportable SCSI Disk Shadow-Set Members

Published for V6.2 and V7.0—I'm planning to publish for V7.1.

V6.2

An HSZ40 Raid-Array Controller provides the capability of an OpenVMS initialized SCSI disk (that is, one with a Files–11 ODS-2 format on it), to be moved between an OpenVMS controlled SCSI bus and an HSZ40 controlled SCSI bus, without reinitializing the disk and losing data. Disks that contain this functionality are called **transportable** disks.

A SCSI disk initialized by the HSZ40 and then subsequently initialized by OpenVMS is called a **nontransportable** disk, and cannot be moved to an OpenVMS controlled SCSI bus without losing data.

OpenVMS Volume Shadowing requires that a SCSI disk support the SCSI commands READ_LONG/WRITE_LONG. These SCSI commands in conjunction with OpenVMS Volume Shadowing are used to handle certain classes of errors as seen under normal volume shadowing operations. SCSI disks that support the READ_LONG/WRITE_LONG capability while connected to an OpenVMS controlled SCSI bus, lose this capability when the transportable disks are moved to the SCSI bus controlled by an HSZ40.

The lack of READ_LONG/WRITE_LONG capability is detected at shadow-set MOUNT time, by the following error:

```
MOUN$_DEVNOFE, device does not support FORCED ERROR handling
```

To correct this problem, specify the MOUNT qualifier /OVERRIDE=NO_FORCED_ERROR at shadow-set MOUNT time.

Note that specifying this MOUNT qualifier may cause shadow-set member SCSI disks to be removed from a shadow set if certain error conditions arise that cannot be corrected.

Digital recommends that HSZ40 nontransportable SCSI disks be used to contain shadow-set members that support READ_LONG/WRITE_LONG functionality, and offer benefits provided by the level of RAID chosen at initialization time.

2.1.2.3.4 SCSI Disk I/O Performance Degradation for KZMSA XMI and Adaptec 1742A Adapters

Published for V6.2 and V7.0—OK to delete for V7.1?????

V6.2

As a result of the SCSI-2 Tagged Command Queuing (TCQ) support in OpenVMS Alpha Version 6.2, Digital has determined that customers with KZMSA XMI to SCSI and Adaptec 1742A adapters might experience a 20 percent SCSI disk I/O performance degradation because TCQ is not implemented for these adapters. The performance degradation is in the area of increased CPU cost per I/O. Customers running at less than maximum CPU utilization under OpenVMS Alpha Version 6.1 might not experience any degradation under OpenVMS Alpha Version 6.2.

Digital does not expect this situation to significantly affect DEC 7000 customers planning to upgrade to DEC 8000 Family systems using KZMSA adapters because the speed of those processors should offset the performance degradation. However, DEC 7000 customers who upgrade to OpenVMS Alpha Version 6.2 will experience the SCSI I/O disk performance degradation.

Digital expects that this will significantly affect existing DEC 2000 Model 300 systems customers that use the Adaptec 1742A SCSI adapter.

2.1.2.4 C Structure Member Names: Passing to Nested C Macros

This remains a restriction. I'm planning to publish for V7.1.

V6.1

The file SYS$LIBRARY:SYS$LIB_C.TLB is a library of C header files that define structures corresponding to the structures defined for MACRO-32 in SYS$LIBRARY:LIB.MLB. These C structure definitions also include simple macros, which allow the use of the same field names in drivers written in C as those used in drivers written in MACRO-32 and BLISS.

A compile-time error results if both of the following occur:

- You pass one of these simple macro names for a structure member as part of a parameter to a C macro
- This macro passes this parameter to another macro

You can avoid this problem by using one of the following methods:

- ''Flattening'' the macro definition (by expanding the nested macros)
- Using C inline functions to replace the nested macros while maintaining the desired modularity
- Introducing additional temporary storage in the outer macro to avoid passing the input arguments directly to the nested macros

2.1.2.5 OpenVMS Alpha System-Code Debugger

I'll verify this info with Richard Bishop and Steve Dipirro.

V6.1

The OpenVMS Alpha System-Code Debugger has the following problems and limitations:

- The CALL command is not currently implemented.
- The SET WATCH command uses nonstatic watchpoints. Because nonstatic watchpoints are slower than static watchpoints, their use is not recommended.
- With the OpenVMS Debugger, if you use the SET BREAK/EXCEPTION command, when the program hits an exception, the user could fix the problem and continue the program. With the system-code debugger, the SET BREAK/EXCEPTION command catches the problem later (just before a bugcheck). Therefore, it is not possible to fix the problem and proceed. The GO command lets the system bugcheck and write the crash dump.
- Setting breakpoints at IF statements in the C programming language does not always work. For example, if you have code like the following and you set a breakpoint at the first line, the system-code debugger may or may not hit the breakpoint (depending on the condition):

```
if (x)
{
        ....
} else
{
          ...
}
```

To avoid this problem, you can set two breakpoints, one in the THEN clause and one in the ELSE clause, or set a breakpoint before the IF statement and then single-step until you reach the IF statement.

- When using the SET MODULE command on some images with the system-code debugger, you may receive an error message similar to the following:

  ```
  %DEBUG-E-INVPD, procedure descriptor at 00028840 is not valid.
  ```

  This procedure descriptor is either in the init section for the image or in a paged-out image section. You can determine this by searching for the above value in the MAP file for the image to which the module belongs.

  If the value is not within the init image section or a pageable section, file a Problem Report and provide the error message along with the MAP file for the image. ♦

2.2 Device Support on OpenVMS VAX Systems

VAX

All of these notes stay because the VAX driver books have been archived.

The following sections contain release notes pertaining to OpenVMS VAX device support.

2.2.1 Documentation Changes and Corrections

This section describes corrections to device support documentation.

2.2.1.1 *OpenVMS VAX Device Support Manual*

V6.1

The following sections describe corrections to the *OpenVMS VAX Device Support Manual*.

2.2.1.1.1 Linking a Device Driver

V6.1

Chapter 12 of the *OpenVMS VAX Device Support Manual*, Version 6.0, describes how to assemble, link, and load a device driver. In Step 3 of the procedure for preparing a driver for loading into the operating system, append the following text to the end of the procedure (following the paragraph that begins: ‘‘The resulting image must . . . ’’):

To produce an image with a symbol table compatible with the System Dump Analyzer (SDA), you must link again; this time, using the UNIVERSAL=* option statement (to include all global symbols and to ensure proper state of the REL bits in the object records). Relink as shown in the following example:

```
$ LINK /NOEXECUTABLE/NOTRACEBACK/NOSYSSHR -
_$     /SYMBOLS=MYDRIVER.EXE,-
_$     /SHARE=DUMMY_FILE_NAME,-
_$     /NOMAP,MYDRIVER1.OBJ,MYDRIVER2.OBJ,-
_$     SYS.STB/SELECTIVE,-
_$     SYS$INPUT/OPTION
_$     BASE=0
_$     UNIVERSAL=*
```

For more information about the Linker, see the *OpenVMS Linker Utility Manual*.

2.2.1.1.2 Device-Register I/O Space: Usage Restrictions

V6.1

Chapter 5 of the *OpenVMS VAX Device Support Manual*, Version 6.0, describes device driver coding and the restrictions on the use of device-register I/O space. The third sentence of the fifth bulleted paragraph in Section 5.2 states that the instructions that refer to UNIBUS adapter registers must use longword context. This is the wrong bus. The sentence should read:

> ''Instructions that refer to MASSBUS adapter registers must use the longword context.''

2.2.1.2 *OpenVMS VAX Device Support Reference Manual*

V6.1

The following sections describe corrections to the *OpenVMS VAX Device Support Reference Manual*.

2.2.1.2.1 COM$DRVDEALMEM Routine Synchronization

V6.1

Chapter 3 of the *OpenVMS VAX Device Support Reference Manual*, Version 6.0, contains a section describing the COM$DRVDEALMEM routine.

At the end of the paragraph under *Synchronization*, add the following sentence:

''If called at IPL$_SYNCH or higher, the routine executes the fork process.''

2.2.1.2.2 CRB Data Structure

V6.1

Chapter 1, Section 1.7, of the *OpenVMS VAX Device Support Reference Manual*, Version 6.0, contains Table 1-8 describing the CRB data structure fields. The description in the table for the CRB$L_INTD field is confusing and needs clarification. Replace the first two sentences in the CRB$L_INTD description as follows:

Field Name	Description
CRB$L_INTD	Portion of the interrupt transfer vector block that stores executable code, driver entry points, and I/O adapter information. This 10-longword area is overlaid with the contents of the interrupt transfer vector block that starts at VEC$L_INTD (offset 16) as described in Section 1.7.1. It contains pointers to the driver's . . .

2.2.1.2.3 SCDRP Data Structure SCSI Flags

V6.1

Chapter 1 of the *OpenVMS VAX Device Support Reference Manual*, Version 6.0, contains a section describing the SCDRP data structure SCSI flags.

In the SCDRP$L_SCSI_FLAGS field description for bit SCDRP$V_LOCK, make the following correction:

Change: SCDRP$VLOCK

To: SCDRP$V_LOCK

2.2.1.2.4 SPI$CONNECT Macro

V6.1

Chapter 2 of the *OpenVMS VAX Device Support Reference Manual*, Version 6.0, contains a section describing the SPI$CONNECT macro.

In the table listing the required inputs, add:

R4 Address of the SPDT

In the values returned in R3, the SPDT$M_CMDQ bit was added to the port capability mask (SPDT$L_PORT_FLAGS). When set, SPDT$M_CMDQ indicates that the port driver supports command queuing I/O.

In the return values table listing R3 and the mask bits (after SPDT$M_LUNS), add:

SPDT$M_CMDQ Supports command queuing I/O

2.2.1.2.5 SPI$GET_CONNECTION_CHAR and SPI$SET_CONNECTION_CHAR Macros

V6.1

Chapter 2 of the *OpenVMS VAX Device Support Reference Manual*, Version 6.0, contains sections describing the SPI$GET_CONNECTION_CHAR and SPI$SET_CONNECTION_CHAR macros. Appended to the macro characteristics buffer is longword #12 for SCSI-2 support.

At the end of the characteristics buffer table in these macro descriptions, add the longword #12 information as follows:

12 SCSI-2 device characteristic status bits. Bits of this longword are defined as follows:

Bit	Description
0	When set, (SCDT$V_SCSI_2) indicates the device connection is SCSI-2 conformant.
1	When set, (SCDT$V_CMDQ) indicates the device connection supports command queuing.

2.2.1.2.6 $EQULST Macro

V6.1

Chapter 2 of the *OpenVMS VAX Device Support Reference Manual*, Version 6.0, contains a section describing the $EQULST macro.

In the parameter description for **symbol,value** insert the phrase *in decimal* as follows:

‘‘ . . . and value specifies in decimal the value of the symbol.’’ ♦

3

Goblin (OpenVMS Alpha V 6.2—1H3) Device Naming Notes

Note

The release notes in this chapter were published for the Goblin release. We are not planning to publish them for Gryphon, but we can discuss this decision at Wednesday's review meeting.

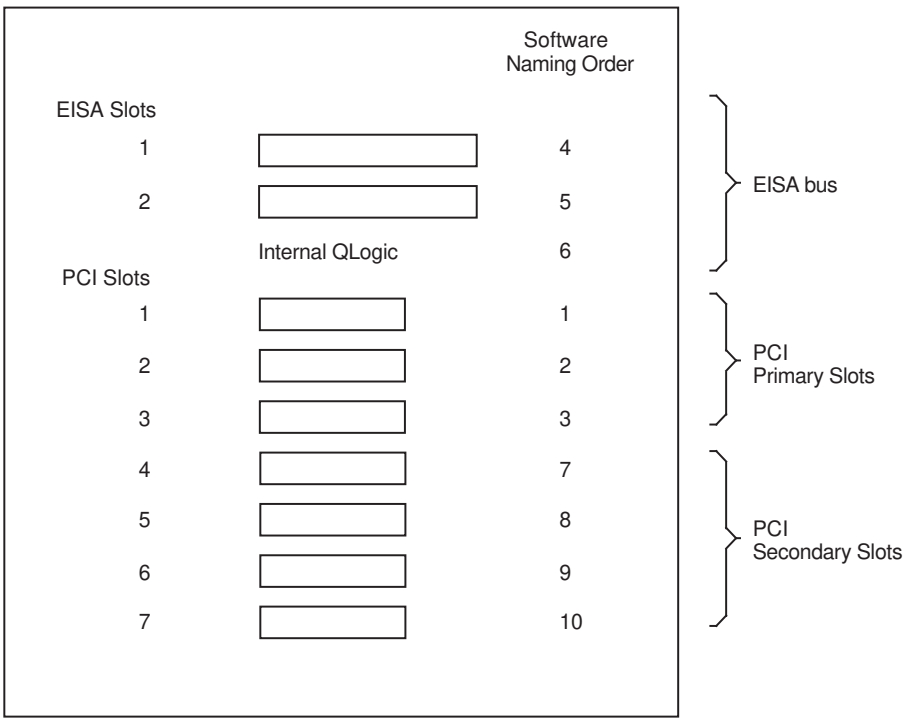
3.1 Device Naming on AlphaServer 1000A Systems

AlphaServer 1000A systems have 7 PCI option slots and 2 EISA option slots, in addition to an integrated QLogic 1020A Fast Wide SCSI controller. The OpenVMS operating system and the console do not always assign the same names to devices in these option slots. In addition, the slot numbering convention used by the console and the OpenVMS operating system does not match the physical slot numbers on the backplane.

The EISA slots are connected using a PCI-to-EISA bridge on the primary PCI bus. The upper three PCI option slots (numbered 1 to 3) are also on the primary PCI bus, while the lower four PCI option slots (numbered 4 to 7) are connected using an integrated PCI-to-PCI bridge. The QLogic 1020A SCSI controller is in a fixed position relative to the option slots; this controller is always configured as the first device behind the integrated PCI-to-PCI bridge, preceding slots 4 to 7. (See Figure 3–1.)

Goblin (OpenVMS Alpha V 6.2—1H3) Device Naming Notes
3.1 Device Naming on AlphaServer 1000A Systems

Figure 3–1 AlphaServer 1000A Hardware Backplane



ZK–8661A–GE

The OpenVMS operating system assigns device names in the following order:

1. Devices on the primary PCI (slots 1 to 3)
2. EISA devices
3. Integrated SCSI controller
4. Devices on the secondary PCI (slots 4 to 7)
5. Devices on PCI-to-PCI bridges in secondary PCI slots
6. Devices on PCI-to-PCI bridges in primary PCI slots

Consider a system with only a KZPSA SCSI controller in slot 1, the first slot on the primary PCI bus. The KZPSA appears as PKA, and the integrated QLogic 1020A appears as PKB. If another KZPSA were added in slot 4, behind the integrated bridge, it would appear as PKC. If yet another KZPSA were added in slot 3, the third slot on the primary PCI bus, the device names would shift; the original KZPSA in slot 1 remains PKA, the new KZPSA in slot 3 becomes PKB, the QLogic becomes PKC, and the KZPSA in slot 4 becomes PKD.

In another example, consider a system with one KZPSA in slot 4. This KZPSA is behind the integrated bridge, behind the QLogic 1020A, and appears as PKB with the QLogic 1020A as PKA. A system with only bridge cards in its option slots, for instance, P1SE bridges with integrated SCSI controllers, would always see the QLogic 1020A as PKA and the SCSI controllers on the P1SE bridges as PKB and so forth.

3.1.1 Restriction for PCI Adapters

As shown in Figure 3–1, PCI slots 4-7 on AlphaServer 1000A systems are connected by an integrated PCI to PCI bridge. Certain PCI adapters, such as the KZPSM, also contain an onboard PCI to PCI bridge. Because the KZPSM is not supported in PCI slots 4-7, you must place that adapter in PCI slots 1, 2, or 3.

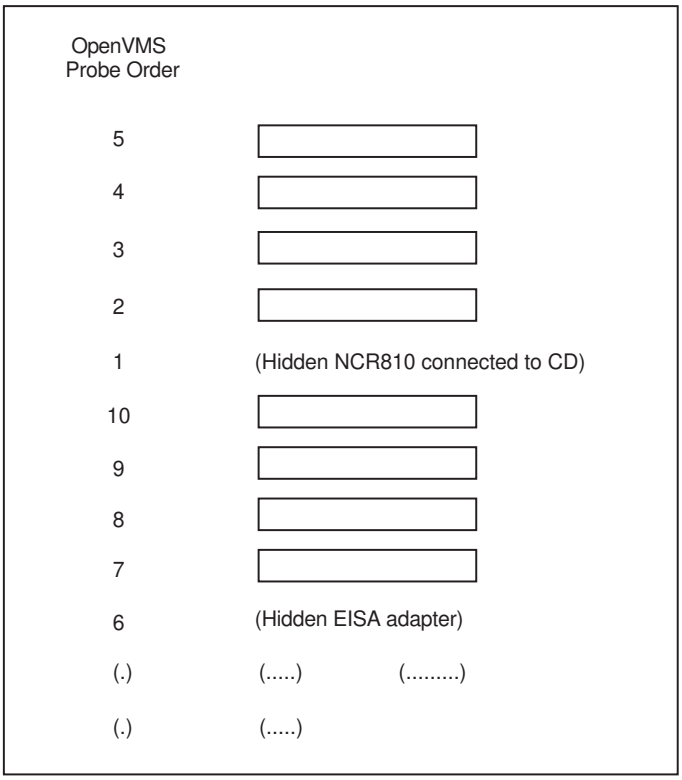
3.2 Device Naming on AlphaServer 4100 Systems

AlphaServer 4100 systems have 8 PCI option slots, 3 of which are shared with EISA option slots. Because the OpenVMS operating system and the console each have their own device naming conventions, the console name of a device will not necessarily match the OpenVMS name of a device. In addition, the slot numbering convention used by the console and the OpenVMS operating system does not match the physical slot order on the backpanel.

AlphaServer 4100 systems also include an integrated NCR810 SCSI controller and an integrated PCI/EISA bridge. The integrated NCR810 SCSI controller is connected directly to the compact disc reader.

The order in which OpenVMS names devices is shown by the OpenVMS Probe Order column in Figure 3–2.

Figure 3–2 AlphaServer 4100 Hardware Backpanel



ZK–8684A–GE

The integrated NCR810 controller (connected to the compact disc reader) will always be assigned the same name by both the console and by OpenVMS because it is always named first. The OpenVMS operating system assigns device names in the following order (using the OpenVMS probe numbers shown in Figure 3–2):

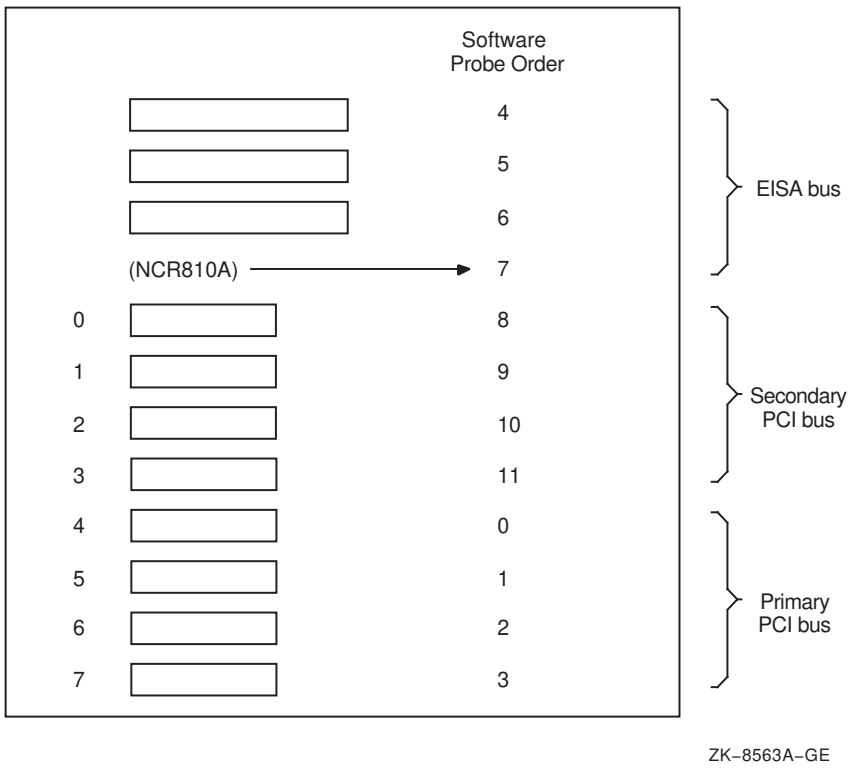
1. Integrated NCR 810 connected to the CD
2. PCI slots 2 through 5
3. Devices on PCI-to-PCI bridges on PCI slots 2 through 5
4. PCI slots 7 through 10
5. EISA slots 7 through 9
6. Devices on PCI-to-PCI bridges on PCI slots 7 through 10

3.3 Device Naming on AlphaServer 2100A Systems

AlphaServer 2100A systems have 8 PCI option slots and 3 EISA option slots, in addition to an integrated NCR810A SCSI controller. The OpenVMS operating system and the console do not always assign the same names to devices in these option slots. In addition, the slot numbering convention used by the console and by the OpenVMS operating system does not match the physical slot numbers on the backplane.

The EISA slots are connected using a PCI-to-EISA bridge on the primary PCI bus. The bottom four PCI option slots (hardware numbers 4 to 7) are also on the primary PCI bus, while the remaining four PCI option slots (the top slots, hardware numbers 0 to 3) are connected using an integrated PCI-to-PCI bridge. The NCR810A SCSI controller is in a fixed position relative to the option slots; the NCR810A controller is always configured as the first device behind the integrated PCI-to-PCI bridge, preceding PCI slots 0 to 3. (See Figure 3–3.)

Figure 3–3 AlphaServer 2100A Hardware Backplane



The OpenVMS operating system assigns device names in the following order:

1. Devices on the primary PCI (slots 4 to 7)
2. EISA devices
3. Integrated NCR810A SCSI controller
4. Devices on the secondary PCI (slots 0 to 3)
5. Devices on PCI-to-PCI bridges in secondary PCI slots
6. Devices on PCI-to-PCI bridges in primary PCI slots

Consider a system with only a KZPSA SCSI controller in slot 4, the first slot on the primary PCI bus. The KZPSA appears as PKA, and the integrated NCR810A appears as PKB. If another KZPSA were added in slot 0, behind the integrated bridge, it would appear as PKC. If yet another KZPSA were added in slot 6, the third slot on the primary PCI bus, the device names would shift; the original KZPSA in slot 4 remains PKA, the new KZPSA in slot 6 becomes PKB, the 810A becomes PKC, and the KZPSA in slot 0 becomes PKD.

In another example, consider a system with one KZPSA in slot 0. This KZPSA is behind the integrated bridge, behind the 810A, and appears as PKB, with the 810A as PKA. A system with only bridge cards in its option slots, for instance, P1SE bridges with integrated SCSI controllers, would always see the 810A as PKA and the SCSI controllers on the P1SEs as PKB and so forth.