1

Device Support on OpenVMS Systems

This chapter contains release notes pertaining to OpenVMS device support on Alpha and VAX systems. For each note, the heading indicates whether it contains Alpha-specific or VAX-specific information.

1.1 Recompiling and Relinking OpenVMS Device Drivers

The following sections contain release notes pertaining to recompiling and relinking OpenVMS device drivers.

1.1.1 OpenVMS Alpha and OpenVMS VAX SCSI Device Drivers

V7.1

OpenVMS Alpha 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 OpenVMS VAX SCSI device drivers from OpenVMS Version 7.0 must be recompiled and relinked to run correctly on OpenVMS Alpha Version 7.1. No source changes are required.

1.1.2 OpenVMS Alpha Device Drivers for Version 7.0

V7.1

OpenVMS Alpha Version 7.0 included significant changes to OpenVMS Alpha privileged interfaces and data structures. As a result of these changes, all device drivers from releases prior to OpenVMS Alpha Version 7.0 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 Version 7.1 Note

OpenVMS Alpha device drivers that were recompiled and relinked for OpenVMS Alpha Version 7.0.

OpenVMS Alpha SCSI device drivers from OpenVMS Version 7.0 must be recompiled and relinked as described in Section 1.1.1.

1.2 Recompiling ICBMs for OpenVMS Alpha Version 7.1

V7.1

Customer-written ICBMs (used to autoconfigure device drivers) from previous releases must be recompiled and relinked to run on OpenVMS Version 7.1.

Recompiling and relinking ICBMs is necessary for this release because of additions to the bus array entry structure.

1.3 Restriction for Third-Party Tape Applications and Non-ANSI Standard Tapes on SCSI Tape Drives

V7.1

Writer's Note

I am revising this note to include the following information; I will send it out for review separately:

- There is a new modifier for IO$_SKIPFILE that provides much better performance on SCSI tape drives that support the new mechanism. Refer to New Features book and the I/O User's Guide.
- When the modifier is used, IO$_SKIPFILE stops at the end of data, rather than double filemarks.
- MTAACP and Backup use the new modifier. The behavior of MTAACP and Backup are unchanged when ANSI-formated tapes are used. They may behave differently when non-ANSI standard tapes are used. If this is a problem, or if you have an application that can use the new IO$_SKIPFILE semantics, but can not be modified to use the new call, refer to the documentation in SYS$ETC:MKSET.TXT.

-Margie

Digital recommends that OpenVMS VAX and Alpha Version 7.1 not be used with any third-party tape applications or tapes that use a non-ANSI standard tape format. This restriction applies only to SCSI tape drives that have the MK device type name in their physical device names.

The performance of the IO$_SKIPFILE QIO function has been significantly improved in OpenVMS Version 7.1 for certain SCSI tape drives. The new IO$_ SKIPFILE implementation has been shown to function correctly with all built-in OpenVMS tape functions such as INIT, MOUNT, BACKUP, and COPY when tapes are formatted according to the ANSI Standard X3.27-1987. This is the default tape standard for OpenVMS.

The new IO$_SKIPFILE implementation differs from the previous implementation in that it does not terminate if two consecutive filemarks are encountered. Instead, it terminates when the end of recorded data is encountered. This change may cause problems when applications that have been coded to expect the old behavior are run with tape drives that support the new SKIPFILE implementation. It may also cause problems for any application that is used with a non-standard tape format.

Digital expects to remove this restriction for the final release of OpenVMS Version 7.1.

1.4 OpenVMS Alpha SYS$MANAGER:ISA_CONFIG.DAT Unsupported in Future Release

V7.1

Support for using the SYS$MANAGER:ISA_CONFIG.DAT file to configure ISA devices will be discontinued in a future release of OpenVMS Alpha. If you use this file, you should convert to using the ISACFG utility from the console, and the new file-based autoconfiguration method for loading device drivers (as described in the *OpenVMS Version 7.1 New Features Manual*).

1.5 Required Change for ISA_CONFIG.DAT on AlphaStation 200/400 Family Systems

V7.1

Customers configuring ISA devices on AlphaStation 200/400 Family systems must change their ISA_CONFIG.DAT file, so that the node information for each device appears at the end of each device description block.

For upgrades from OpenVMS Version 6.2 or 7.0 systems, this change must be made before the upgrade.

For example, prior to OpenVMS Version 7.1, the following device description block:

```
[AUA0]
 NAME=AU
 NODE=3
 DRIVER=SYS$MSBDRIVER
 IRQ=9
 DMA=(0,1)
 PORT=(388:4,530:8)
```

would be changed as follows:

```
[AUA0]
 NAME=AU
 DRIVER=SYS$MSBDRIVER
 IRQ=9
 DMA=(0,1)
 PORT=(388:4,530:8)
 NODE=3
```

1.6 Naming Serial Line Devices on Alpha Systems

V7.1

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, 2100, and 4100 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 from that 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.

1.7 Behavior of Wide SCSI Drives on Qlogic

Note

This note will be supplied by Sue Sommer after I return from DECUS.

-Margie

1.8 Memory Holes on AlphaServer 4100 Systems

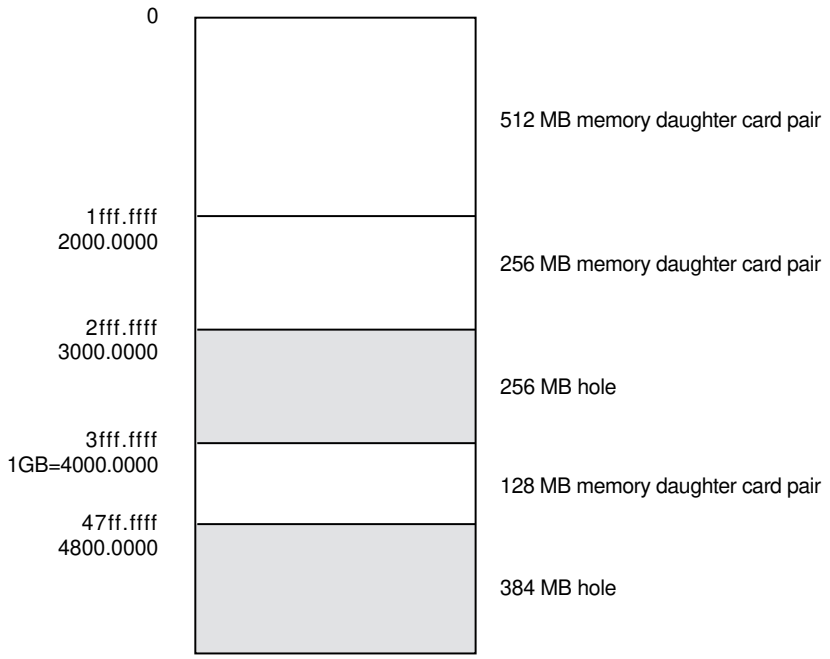
V7.1

Physical memory holes might exist on AlphaServer 4100 systems. As illustrated in the example diagram in this section, there are three different sizes of memory daughter card pairs: 512 MB, 256 MB and 128 MB. In accordance with AlphaServer 4100 systems configuration rules, memory card pairs must be arranged in descending order of size.

The AlphaServer 4100 hardware reads the first set of memory daughter cards and assumes that any memory card pairs that follow are the same size. Memory holes occur because memory card pairs following the first set of cards read by the hardware might not be the same size. As shown in the following diagram, the hole at 3000.0000 must be dealt with by OpenVMS. The hole at 4800.0000 is at the top of the address space and can be ignored by OpenVMS.

Note

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



Note that this configuration impacts the algorithm used to determine whether a driver needs to use map registers. In releases prior to OpenVMS Alpha Version 7.1, device 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$NODE_DATA to pages and compare the size with **mmg$gl_memsize**, which contains the number of pages in physical memory.
3. If **mmg$gl_memsize** is greater than the size returned from IOC$NODE_ DATA, use map registers; otherwise, use the direct DMA window.

The **mmg$gl_memsize** global cell does not contain the memory hole. As a result, the system has only 7/8 GB of memory, but according to the algorithm in releases prior to OpenVMS Alpha Version 7.1, it appears that the device can use the direct DMA window. Yet there is 128 MB of memory beyond the 1 GB border, which requires that the drivers use map registers. To eliminate this problem, drivers using the algorithm in releases prior to OpenVMS Alpha Version 7.1 must substitute it with the following algorithm:

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$NODE_DATA to pages by dividing the number of bytes by the contents of **mmg$gl_page_size**. For example:

```
int dma_size;
int pages;

status = IOC$NODE_DATA (crb, IOC$K_DIRECT_DMA_SIZE, &dma_size);
        /* dma_size contains the number of megabytes.
         * convert number of megabytes to bytes.
         */
dma_size = dma_size * (1024 * 1024);
        /* Convert number of bytes to number of pages by
         *  dividing by number of bytes per page.
         */
pages = dma_size / MMG$GL_PAGE_SIZE;
```

3. Compare the resulting number of pages with **mmg$gl_maxpfn + 1**.
4. If **mmg$gl_maxpfn + 1** is greater than the size returned from IOC$NODE_ DATA, use map registers; otherwise use the direct DMA window.

1.9 SYS$MSBDRIVER Removed from OpenVMS Alpha Distribution

V7.0

The driver for the Microsoft Windows Sound System ISA sound card (MSB), SYS$MSBDRIVER, has been removed from the OpenVMS Alpha 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.)

1.10 Device IPL Setup for OpenVMS Alpha Drivers

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.

1.11 AlphaStation 255: PCI Configuration Restriction

V7.1

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

PCI slot 0 is the lowest physical PCI option slot on AlphaStation 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 255 series systems support a maximum of two PCI option cards, configured in slot 1 and slot 2.

1.12 Configuration Restriction for RZ25M and RZ26N Disk Drives on OpenVMS Alpha

V7.1

During the testing of Digital supported SCSI disk drives on configurations with DWZZAs and long differential SCSI buses, two drives (RZ25M and RZ26N) were found to have bus phase problems. For this reason, these drives are not recommended for use in configurations where the differential bus length connecting DWZZAs equals or exceeds 20 meters.

This restriction applies only to the RZ25M and RZ26N drives. All other disk drives, listed as supported in the OpenVMS SPD, may be used in configurations to the full bus lengths of the SCSI-2 specification.

1.13 SCSI Controller Restriction on AlphaServer 2100 Systems

The Adaptec 1740/1742 SCSI controller (PB2HA–SA) is not supported on AlphaServer 2100 systems having more than 1 gigabyte (GB) of memory. If the controller is connected to such a system, the following message appears on the operator's console:

```
%PKJDRVR-E- The direct DMA window does not map all of memory. Port is going OFF)
```

1.14 OpenVMS Alpha SCSI Firmware Support

The following sections relate to SCSI firmware support.

1.14.1 Updating Firmware in SCSI Configurations

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.

1.14.2 Required Firmware Support for RZ26N and RZ28M Disks

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.

1.14.3 SCSI Firmware Update Instructions

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 1.14.4 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 1.14.5 to update the disk's firmware.

1.14.4 Firmware Revision Level 442D Requirements

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 1–1 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

1.14.5 442D Firmware Installation Procedure

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 1–1 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.
```

1.15 OpenVMS Alpha SCSI Port and Class Drivers

V6.2

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

1.15.1 Add-On SCSI Adapters

V6.2

Version 6.2 and later of OpenVMS Alpha 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.

1.15.2 HSZ40 and Transportable SCSI Disk Shadow-Set Members

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.

1.16 OpenVMS VAX Device Support Documentation Corrections

This section describes corrections to OpenVMS VAX device support documentation.

1.16.1 *OpenVMS VAX Device Support Manual*

V6.1

The following sections describe corrections to the *OpenVMS VAX Device Support Manual*. (This manual has been archived but is available in PostScript and DECW$BOOK (Bookreader) formats on the OpenVMS Documentation CD–ROM. A printed book can be ordered through DECdirect (800–354–4825).)

1.16.1.1 Linking a Device Driver

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*.

1.16.1.2 Device-Register I/O Space: Usage Restrictions

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.''

1.16.2 *OpenVMS VAX Device Support Reference Manual*

V6.1

The following sections describe corrections to the *OpenVMS VAX Device Support Reference Manual*. (This manual has been archived but is available in PostScript and DECW$BOOK (Bookreader) formats on the OpenVMS Documentation CD–ROM. A printed book can be ordered through DECdirect (800–354–4825).)

1.16.2.1 COM$DRVDEALMEM Routine Synchronization

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.''

1.16.2.2 CRB Data Structure

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 . . .

1.16.2.3 SCDRP Data Structure SCSI Flags

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

1.16.2.4 SPI$CONNECT Macro

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

1.16.2.5 SPI$GET_CONNECTION_CHAR and SPI$SET_CONNECTION_CHAR Macros

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.

1.16.2.6 $EQULST Macro

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.’’