VAX Disk Striping Driver for VMS
Guide to Operations

Order Number: AA–NY13B–TE

April 1991

This manual provides use and management information for the VAX Disk Striping Driver for VMS software. Disk striping converts individual I/O requests into multiple I/O requests that are then processed concurrently on multiple disk drives. This results in an increased data rate and a reduction in the effective seek distance per megabyte.

Digital Equipment Corporation
Maynard, Massachusetts

First Edition, December 1989
Second Edition, December 1990

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Any software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013.

FCC NOTICE: The Equipment described in this manual generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference, in which case the user at his own expense may be required to take measures to correct the interference.

Copyright ©1990 Digital Equipment Corporation

All Rights Reserved.
Printed in U.S.A.

The Readers Comment card, at the end of the hardcopy version of this document, requests the user's critical evaluation in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: CI, DEC, DECUS, KDA, KDM70, MASSBUS, RA, RA82, RA90, UNIBUS, MicroVAX, MSCP, PDP, UDA, VAX, VAX DOCUMENT, VAX Performance Advisor, VAX Volume Shadowing, VAXcluster, VAXsimPLUS, VMS, and the DIGITAL logo.

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

PREFACE vii

CHAPTER 1 PRODUCT OVERVIEW 1–1

1.1 HOW DOES STRIPING WORK? 1–2

1.2 WHAT ARE THE BENEFITS? 1–4
1.2.1 High Data Transfer Rates 1–4
1.2.2 Disk Load Balancing 1–5
1.2.3 High Throughput 1–6
1.2.4 Reliability 1–6

CHAPTER 2 STRIPESET CONFIGURATION AND MANAGEMENT 2–1

2.1 CREATING A STRIPESET ON A SINGLE CPU 2–1
2.1.1 Initializing Member Units 2–1
2.1.2 Binding Member Units into a Stripeset 2–2
2.1.3 Creating a VMS File System on the Stripeset 2–3
2.1.4 Mounting a VMS File System on the Stripeset 2–3
2.1.4.1 Example–Single CPU Stripeset Creation • 2–4

2.2 THE VAXCLUSTER ENVIRONMENT 2–5
2.2.1 Planning For Your Stripeset Configuration 2–5
2.2.1.1 Do I Need to MSCP Serve? • 2–5
2.2.1.2 Choosing the MSCP Serving Strategy • 2–5
2.2.2 Allocation Class 2–8
2.2.2.1 What Is It and Why Is It Important? • 2–8
2.2.2.2 How Do I Set the Allocation Class? • 2–8
2.2.3 Enabling the MSCP Server 2–9
2.2.4 Creating a Stripeset on a VAXcluster System 2–10
2.2.4.1 Striping on a VAXcluster System Prior to VMS Version 5.4 Software • 2–11

2.3 USING SHADOWED DISKS 2–13
2.3.1 Combining Disk Striping with Volume Shadowing in a VAXcluster System 2–14

Contents

2.3.2 Data Protection Using VAXsimPLUS Software 2–16

2.4 UNBINDING THE STRIPESET 2–16

2.5 STRIPESET CONFIGURATION RESTRICTIONS 2–17

2.6 MANAGING STRIPESETS LARGER THAN 8.5 GB 2–17
2.6.1 Determining the Stripeset's Maximum Transfer Size 2–17
2.6.2 Maintaining a Stripeset 2–19

CHAPTER 3 INSTALLATION GUIDE 3–1

3.1 PREINSTALLATION PLANNING 3–1
3.1.1 Configuration Planning Checklist 3–1
3.1.2 Preinstallation Requirements 3–2
3.1.3 Preinstallation Restrictions 3–2
3.1.4 Time Needed for Installation 3–2
3.1.5 Preinstallation Procedure 3–2
3.1.6 Adjusting SYSGEN Parameters 3–3
3.1.6.1 Detecting SYSGEN Problems • 3–4
3.1.6.2 Using AUTOGEN • 3–4
3.1.7 Upgrading Disk Striping Software From Version 1.0 to Version 2.0 3–5
3.1.8 Registering the License Product Authorization Key 3–5

3.2 INSTALLING STDRIVER 3–6

3.3 POST-INSTALLATION 3–7
3.3.1 After Installing Disk Striping Software 3–7
3.3.2 Startup File Modifications 3–8

3.4 FILES ADDED TO SYSTEM 3–9
3.4.1 Temporary Files 3–9
3.4.2 Permanent Files 3–9

3.5 VERIFYING THE INSTALLATION 3–9

CHAPTER 4 STRIPESET COMMANDS 4–1
STRIPE BIND 4–3
STRIPE INITIALIZE 4–6
STRIPE LIST 4–8
STRIPE UNBIND 4–10

CHAPTER 5 STRIPE LIST COMMAND USAGE 5–1

5.1 THE STRIPE LIST COMMAND OUTPUT 5–1
5.1.1 STRIPE LIST Fields 5–1

5.2 QIO INTERFACE FOR IO$_STRIPE_LIST 5–3

CHAPTER 6 STRIPING INTERNALS 6–1

6.1 STRIPESET LOGICAL BLOCK MAPPING 6–1
6.1.1 Stripeset Member Structure 6–1
6.1.2 Stripeset Virtual Disk Structure 6–2

6.2 STRIPESET DATA FLOW 6–3
6.2.1 Small Transfers 6–4
6.2.2 Medium Size Transfers 6–4
6.2.3 Large Transfers 6–5

APPENDIX A STRIPE COMMAND ERROR MESSAGES A–1

A.1 MESSAGE FORMAT A–1

A.2 STRIPE MESSAGES A–2

GLOSSARY Glossary–1

Contents

INDEX

EXAMPLES

2–1 Example of Creating a Stripeset on a Single CPU 2–4
2–2 Example—Creating a Stripeset in a VAXcluster Environment 2–12
3–1 Sample Installation 3–6
3–2 Sample Verification 3–10

FIGURES

1–1 Disk Striping Combines Multiple Physical Disks into a Single Virtual Disk 1–2
1–2 Mapping Stripeset LBNs to Member Volume LBNs 1–3
2–1 Striping MSCP Served Disks on a VAXcluster System 2–6
2–2 MSCP Serving the Stripeset Virtual Device on a VAXcluster System 2–7
2–3 Striping Shadow Sets 2–13
2–4 Striping Shadow Sets in a VAXcluster Environment 2–15
6–1 Stripeset Member Logical Block Space 6–2
6–2 Mapping Stripeset Logical Blocks to Member Logical Blocks 6–3

TABLES

1–1 Load Balancing 1–5
2–1 STRIPE INITIALIZE Command and Functions 2–1
2–2 STRIPE BIND Command and Functions 2–2
2–3 Creating a VMS File System on the Stripeset 2–3
2–4 Mounting a VMS File System on the Stripeset 2–3
2–5 MSCP Serving the Stripeset Virtual Device vs. Serving the Stripeset Members 2–5
2–6 Creating a Version 2.0 Stripeset on a VMS Version 5.4 (or later) VAXcluster System 2–10
2–7 Creating a Stripeset on a VMS Version 5.3 (or earlier) VAXcluster with VAX Disk Striping Driver Version 2.0 Software 2–11
2–8 STRIPE UNBIND Command Example and Results 2–16
3–1 Preinstallation Procedures 3–2
5–1 IO$_STRIPE_LIST Item Codes 5–4

Preface

Purpose of This Guide

The *VAX Disk Striping Driver for VMS Guide to Operations* provides usage and management information for the VAX Disk Striping Driver for VMS, Version 2.0 software. This document also contains installation procedures, using the DCL command interface, and a listing of system messages that are unique to disk striping activities.

Intended Audience

The *VAX Disk Striping Driver for VMS, Guide to Operations* is addressed to two major audiences:

- The system administrator who installs the driver software and configures and manages the stripesets.
- The user, who is either a programmer developing applications using disk striping, or an operator running those applications.

Structure of This Manual

This manual consists of the following chapters and appendixes:

- Chapter 1, Product Overview, provides an overview of Disk Striping and describes the product.
- Chapter 2, Stripeset Configuration and Management, includes configuration planning information for system administrators and provides procedures to create stripesets in both single CPU and VAXcluster environments.
- Chapter 3, Installation Guide, provides procedures for the installation and verification of the disk striping software.
- Chapter 4, Stripeset Commands, contains the STRIPE commands in DCL Dictionary format.
- Chapter 5, STRIPE LIST Command Usage, describes the STRIPE LIST command output and the IO$_STRIPE_LIST program interface for application developers.
- Chapter 6, Striping Internals, describes how the VAX Disk Striping Driver software functions. This section provides a deeper level of technical information for the system administrator implementing striping.
- Appendix A, STRIPE Command Error Messages, lists system messages unique to disk striping.

- The Glossary defines terms used in this document.

Referenced Documents

Order numbers for the following referenced documents can be obtained from your Digital Sales Representative.

- *VAX Disk Striping Driver for VMS, Guide to Performance and Tuning*
- *VAX Volume Shadowing Manual*
- *VMS Backup Manual*
- *VMS DCL Dictionary*
- *VMS Guide to Performance Management*
- *VMS License Management Utility Manual* (part of the VMS V5.0 Base Documentation Set)
- *VMS Mount Utility Manual*
- *VMS Performance Advisor User's Guide*
- *VMS SYSGEN Utility Guide*
- *VMS VAXcluster Manual*

Conventions

The following conventions are used in this manual:

Convention	Meaning
[*parameter*]	In command formats, brackets indicate optional parameters. When you enter the optional parameters, do not enter the brackets.
$ SHOW TIME 21-JUN-1989 12:14:52	Command examples show all output lines or prompting characters that the system prints; user input (what you enter) is shown in uppercase type.
italic text	In examples and messages, italic text represents input that is unique for each system. In other areas, italic text is used for emphasis.
boldface text	Boldface text is used for emphasis.
xxxxx	A key name enclosed in a box indicates that you should press that key on the keyboard (for example, Return or Help).
Ctrl/X	A sequence such as Ctrl/X indicates that you need to press Ctrl and a character key simultaneously (for example, Ctrl W or Ctrl O).

Convention	Meaning
data 1 . . . data *n*	In examples, a vertical ellipsis represents the omission of data that the system displays in response to a command or data that a user enters.
file-spec[, . . .]	In command formats, a horizontal ellipsis indicates that the preceding items can be repeated one or more times.

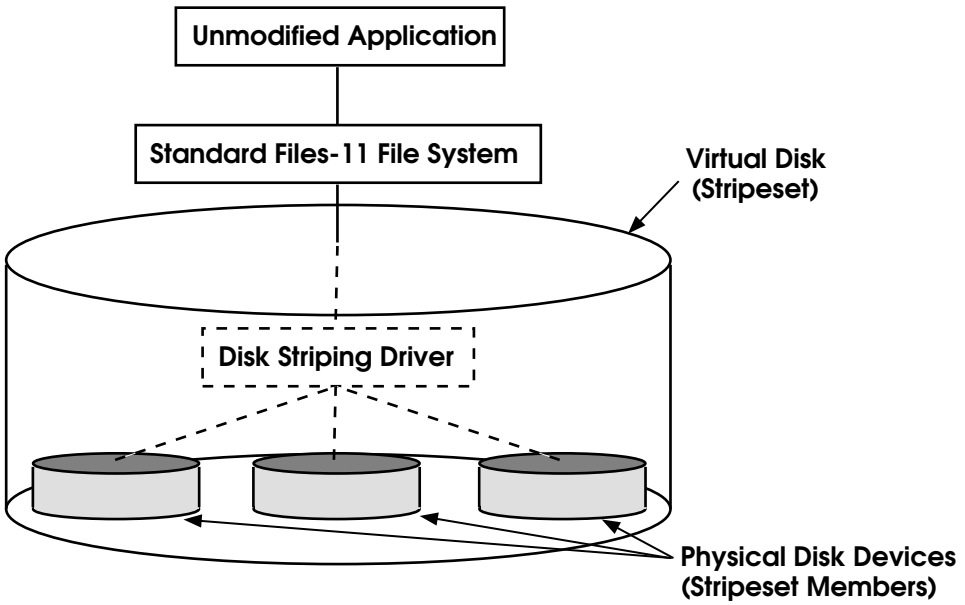
1 Product Overview

The VAX Disk Striping Driver for VMS is software that logically represents two or more identical disks as a single virtual disk. This is done by a device driver called STDRIVER. VMS compatible applications, utilities, databases, and file systems need not be modified to use striping. Figure 1–1, Disk Striping Combines Multiple Physical Disks into a Single Virtual Disk, illustrates this virtual disk.

VAX Disk Striping Driver Version 2.0 fully supports VAXcluster systems. Unlike Version 1.0, a stripeset may be accessed directly from each VAXcluster node which has direct access to the physical disk devices comprising the stripeset. A stripeset may also be MSCP served to VAXcluster nodes that do not have direct access to these physical disks.

Disk striping may also be combined with VAX Volume Shadowing software to form virtual disks that have both high fault tolerance and high throughput. VAXsimPLUS software may also be used to predict physical disk failures within a stripeset and to automatically react to pending failures.

Figure 1–1 Disk Striping Combines Multiple Physical Disks into a Single Virtual Disk



1.1 How Does Striping Work?

Disk striping interleaves multiple disks to improve I/O performance in the same way that high-performance memory systems interleave memory banks. In each case, data requests are decomposed and distributed to physical storage components that process those requests in parallel. This section explains how that is done by the disk striping driver.

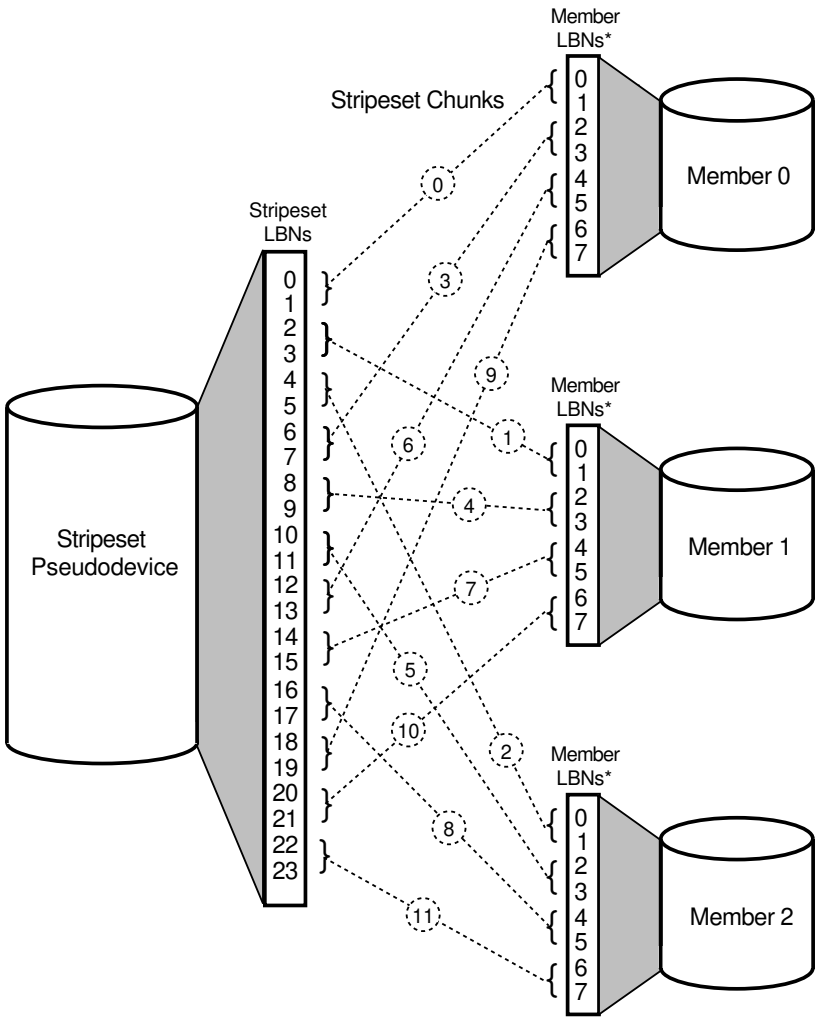
Each disk device appears to the VMS software as an array of logical blocks[1]. A striped virtual disk, or stripeset, consisting of three physical disk devices or members, also appears to the VMS software as an array of logical blocks, but one that is approximately three times larger than each member. The STDRIVER presents the illusion that the combined stripeset members are one disk.

The logical block numbers (LBNs) of the stripeset map *onto* the logical blocks of the stripeset members. That is, each logical block of the stripeset has a corresponding logical block on one of the stripeset members.

[1] Each logical block consists of 512 bytes of storage.

The mapping between stripeset logical blocks and the member logical blocks is determined when the stripeset is initialized and remains fixed during the lifetime of the stripeset. It is controlled by a single parameter called the *chunk size*. Figure 1–2, Mapping Stripeset LBNs to Member Volume LBNs illustrates the mapping of a 3-member stripeset with a chunk size of two.

Figure 1–2 Mapping Stripeset LBNs to Member Volume LBNs



* These LBNs do not take into account the location of the overhead file system.

MR–3226–RA

Each I/O request issued by a VMS application, utility, database, or file system either reads, writes, or erases an arbitrary group of consecutive logical blocks on a disk device. When a large I/O request is issued to a 3-member stripeset, the disk striping driver will decompose the I/O request into three independent I/O requests, one to each member according to the stripeset mapping. Because the three I/O requests are satisfied

concurrently, the time required to transfer the requested data can be reduced by as much as a factor of three.

When a small I/O request is issued to a 3-member stripeset, it is likely that the disk striping driver will not need to decompose the I/O request and will instead deliver it to a single member according to the stripeset mapping. Performance improvements may still result due to load-balancing between the members.

For a more detailed description of striping principles of operation, please refer to Chapter 6, Striping Internals.

1.2 What Are the Benefits?

The purpose of disk striping is to relieve I/O performance problems.

There are three types of I/O performance problems that disk striping alleviates. These problems are most apparent in:

1. applications that require very high data transfer rates,
2. systems in which static disk load-balancing is ineffective, and
3. applications that are prohibited from distributing data across multiple disks, but need higher throughput than is available from a single disk.

Other I/O performance problems may be better addressed with solid-state disk, RAM disk cache, and/or faster disk devices. Similarly, virtual disk devices can be created using standard VMS bound volume sets; however, the load-balancing benefit will not be realized without also implementing the VAX Disk Striping Driver software.

For a comprehensive discussion of the performance benefits of striping, please refer to the *VAX Disk Striping Driver Guide to Performance and Tuning*.

1.2.1 High Data Transfer Rates

Disk striping is ideal for applications that require data transfer rates faster than those available from a single disk drive. A single disk drive is not able to transfer data faster than its spiral transfer rate, which ranges between 1 and 2 MB per second in today's market. Disk striping is able to achieve higher data transfer rates because the striping driver is able to transfer data from the stripeset members in *parallel*.

In order for an application to achieve high transfer rates, the application must issue large (greater than 16 KB) I/O requests. Usually such applications will:

- read or write files sequentially,
- keep multiple I/O requests outstanding simultaneously,
- keep files contiguous, and
- align I/O requests with physical disk track boundaries.

When using disk striping to achieve high transfer rates, it is important to carefully configure the I/O subsystem hardware so that the hardware will not limit the data transfer rate. For further information on this, see the *VAX Disk Striping Driver Guide to Performance and Tuning*.

Applications that require high transfer rates include, but are not limited to:

- image processing,
- seismic data analysis,
- satellite data reception, and
- finite element analysis.

1.2.2 Disk Load Balancing

In most cases, disk striping *automatically* balances the I/O load across the stripeset members.

Application performance can often be improved by manually placing busy files on separate disks so that no disk has an excessively long I/O wait queue. Aside from the effort required, such *manual* disk load balancing is limited because the files that are busy today may not be busy tomorrow.

By striping a very busy disk with others that are not, the I/O load can be automatically balanced across the members of the stripeset, thus average performance can be improved. Disk striping results in load balancing because *portions* of each file are stored on each member of a stripeset, so the activity within a single busy file tends to be distributed evenly across the stripeset members.

Table 1–1, Load Balancing, shows a real example of load balancing due to striping. By changing the bound volume set to a stripeset, the average I/O response time for this application dropped from 35.3 to 22 milliseconds.

Table 1–1 Load Balancing

	Bound Volume Set			Stripeset		
Members	**Queue Length**	**I/O Rate**	**Response Time (ms)**	**Queue Length**	**I/O Rate**	**Response Time (ms)**
DUA1	0.1	3.4	20	0.1	6.7	22
DUA2	0.2	7.4	32	0.2	7.2	22
DUA3	0.4	7.4	56	0.2	7.2	22
DUA4	0.2	7.5	25	0.2	7.2	22
Mean			**35.3**			**22**

For systems to take advantage of the load balancing effects of striping, there must be enough parallel I/O streams so that multiple I/O requests

are outstanding to the stripeset simultaneously. If I/O requests are issued one at a time, the parallel capabilities of the stripeset will not be used because only one member will ever be busy at a time.

1.2.3 High Throughput

For one reason or another, a given application may not be able to distribute data over multiple disks. In some cases, the application is designed to place all of its files on one disk, and because the application is purchased, it is not possible to change the application so that each file is able to reside on an independent disk. In other cases, a single file is extremely busy. Disk striping may be useful if either of these conditions result in poor performance due to an overutilized disk device.

A stripeset of three disks, each of which is capable of 40 I/O requests per second, will be capable of approximately 120 I/O requests per second (3 $\times$ 40 = 120). This occurs because the disk striping driver will tend to distribute these I/O requests to separate stripeset members.

As with load balancing, in order to take advantage of the increased throughput of a stripeset, an application must issue I/O requests so more than one I/O request is simultaneously outstanding. Otherwise, the striping driver will be unable to dispatch requests to members so they complete concurrently.

1.2.4 Reliability

When using a stripeset, it is important to consider the reliability of the stripeset virtual disk. Because portions of each file on the stripeset volume are distributed across all stripeset members, the stripeset *virtual disk* fails when any one of its *member disks* fail.

Measures that *may* be taken to increase the reliability of a stripeset virtual disk include:

- using VAX Volume Shadowing software to improve stripeset member reliability
- using VAXsimPLUS software and its autocopy feature to improve stripeset member reliability
- using high-reliability disk drives for stripeset members
- backing up the stripeset volume frequently, and
- using the stripeset volume for temporary files that can be recreated easily.

2 Stripeset Configuration and Management

This chapter provides detailed instructions for configuring stripesets to best meet your system requirements. These include creating stripesets on single CPU and VAXcluster environments, determining your best MSCP serving strategy, and using the VAX Volume Shadowing software and allocation classes with your stripeset.

Section 2.2 guides you through creating a stripeset for your VAXcluster environment. Because creating a stripeset in a VAXcluster environment is complex, you may want to read Section 2.1 first to develop a basic understanding of how stripesets are created.

2.1 Creating a Stripeset on a Single CPU

This section provides the commands necessary to create a stripeset in a single CPU environment.

2.1.1 Initializing Member Units

To define a stripeset virtual device, you must first prepare the individual member units using the STRIPE INITIALIZE command as shown in the following example. **Doing this will delete all present data on the disk**. If you need this data, you must do an image backup[1] prior to entering the commands in Table 2–1.

Table 2–1 STRIPE INITIALIZE Command and Functions

Command	Result
`$ STRIPE INITIALIZE DUA1,DUA2 TOM,DICK`	• Formats stripeset members and writes data describing the stripeset on each member unit.

This command produces, in this order, the following results:

1. Divides each member disk into three partitions:
   - overhead file system partition,
   - data partition, and
   - RCT partition.
2. Initializes overhead file system.
3. Writes stripeset metadata into STRIPE$VOL.SYS within each overhead file system.

[1] See VMS Backup Manual for more information about performing BACKUP operations.

4 Mounts overhead file system, but these members are not ready for use.

Chunk size is set to the default (1 track). For information about overriding the default, refer to the *VAX Disk Striping Driver Guide to Performance and Tuning*.

2.1.2 Binding Member Units into a Stripeset

The next step in creating a stripeset is to bind the units you have just initialized into a stripeset virtual device using the STRIPE BIND command. At this time, you may associate a systemwide logical name with the virtual device. Table 2–2 shows an example of a STRIPE BIND command.

Table 2–2 STRIPE BIND Command and Functions

Command	Result
`$ STRIPE BIND DUA1,DUA2 DISK$STRIPE`	• Binds the two member units (DUA1 and DUA2) into a stripeset virtual device.

This command produces the the following results:

1 Checks to confirm that the stripeset metadata, written on each member unit by the STRIPE INITIALIZE command, forms a consistent stripeset.

2 Creates a virtual device with the name STA*xx*, which maps member units.

3 Assigns the stripeset virtual device the system logical name of DISK$STRIPE.

Note: Typically, a STRIPE BIND command is placed in the STRIPE$SYSTARTUP.COM file.

2.1.3 Creating a VMS File System on the Stripeset

For a new stripeset, binding results in an empty virtual disk. You will need to initialize that disk.

Table 2–3 Creating a VMS File System on the Stripeset

Command	Result
`$ INITIALIZE/CLUSTER_SIZE=10 DISK$STRIPE STRIPE_VOL`	• Creates a Files-11 file system on the stripeset virtual disk[1].

[1]Refer to the *VMS DCL Dictionary* for more information about the INITIALIZE command.

A stripeset virtual disk may be so large that the INITIALIZE command default CLUSTER_SIZE qualifier will result in an error. To avoid this, specify a CLUSTER_SIZE in the INITIALIZE command, as in Table 2–3.

To calculate the minimum cluster size, divide the stripeset size (in blocks) by the product of 255 $\times$ 4096, and round up.

$$\frac{number\ of\ members \times total\ LBNs\ per\ member}{255 \times 4096}$$

For example, four RA90 disk drives would be: $\frac{\lceil 4 \times 2376153 \rceil}{255 \times 4096} = \lceil 9.1 \rceil = 10$

2.1.4 Mounting a VMS File System on the Stripeset

To make the stripeset available for use, you must enter the command in Table 2–4.

Table 2–4 Mounting a VMS File System on the Stripeset

Command	Result
`$ MOUNT/SYSTEM DISK$STRIPE STRIPE_VOL`	• Makes the stripeset available for use[1].

[1]Refer to the *VMS Mount Utility Manual* for more information on the MOUNT command.

2.1.4.1 Example–Single CPU Stripeset Creation

Example 2–1 shows how the stripeset DISK$STRIPE on a single CPU is created.

Example 2–1 Example of Creating a Stripeset on a Single CPU

```
$ STRIPE INITIALIZE DUA1,DUA2 ST1,ST2 Return

%STRIPE-I-VOLID, unit _$1$DUA1: contains volume STR$M91C0F27
%STRIPE-I-VOLID, unit _$1$DUA2: contains volume STR$M2C4BB72

Proceed to initialize specified units? [Y/N] (N) : yes Return

%STRIPE-I-INIT, initializing member unit DUA1 as volume ST1
%STRIPE-I-INIT, initializing member unit DUA2 as volume ST2
%STRIPE-I-MOUNTING, mounting volume ST1 on unit DUA1
%MOUNT-I-MOUNTED, ST1          mounted on _$1$DUA1: ()
%STRIPE-I-MOUNTING, mounting volume ST2 on unit DUA2
%MOUNT-I-MOUNTED, ST2          mounted on _$1$DUA2: ()

$ STRIPE BIND DUA1,DUA2 DISK$STRIPE Return

%STRIPE-S-BOUNDLOG, stripe set STA9998: defined as DISK$STRIPE
%STRIPE-I-CLUSTER_SUPPORT, DEVICE  CLUSTER WIDE AFTER BINDING ON EACH NODE
%STRIPE-I-WSYNDEQ, Waiting for synch-lock to dequeue

$ INITIALIZE DISK$STRIPE 2_MEMBER Return

$ MOUNT/SYSTEM DISK$STRIPE 2_MEMBER Return

%MOUNT-I-MOUNTED, 2_MEMBER     mounted on _STA9998:

$
```

2.2 The VAXcluster Environment

This section provides information and commands required for successful implementation of stripesets on a VAXcluster system. There are many variables to consider when managing a VAXcluster system. We have provided pertinent information to help you determine the most efficient way to create stripesets that will best meet your needs.

If you are working with a VAXcluster system, we recommend you carefully read all of Section 2.2 before you attempt to create the stripeset, otherwise, critical information may be missed. A sample procedure has been included in Table 2–6.

2.2.1 Planning For Your Stripeset Configuration

Digital recommends planning your configuration changes carefully. This will minimize the number of reboots required to optimize striping for your system.

2.2.1.1 Do I Need to MSCP Serve?

If you have VAXcluster nodes that do not have direct access to the stripeset members and you want access to the stripeset virtual disk from those nodes, then you must use the VMS MSCP server.

2.2.1.2 Choosing the MSCP Serving Strategy

Suppose you want a VAXcluster node, without *local access* to the stripeset members, to have access to the stripeset virtual device.

You can achieve this in one of two ways:

1. MSCP serve the stripeset *virtual device*, or
2. MSCP serve the stripeset *members* and bind the stripeset on the remote node.

Table 2–5 lists the advantages and disadvantages for each of these choices.

Table 2–5 MSCP Serving the Stripeset Virtual Device vs. Serving the Stripeset Members

Serving the	Advantages	Disadvantages
Virtual Device	Less MSCP overhead	Different paths through SYS$STARTUP file for served vs. nonserved nodes.
Members	Common path for all nodes through SYS$STARTUP file.	More MSCP overhead.

Figures 2–1 and 2–2 illustrate what the configurations look like for each of the previous choices.

Figure 2–1 Striping MSCP Served Disks on a VAXcluster System

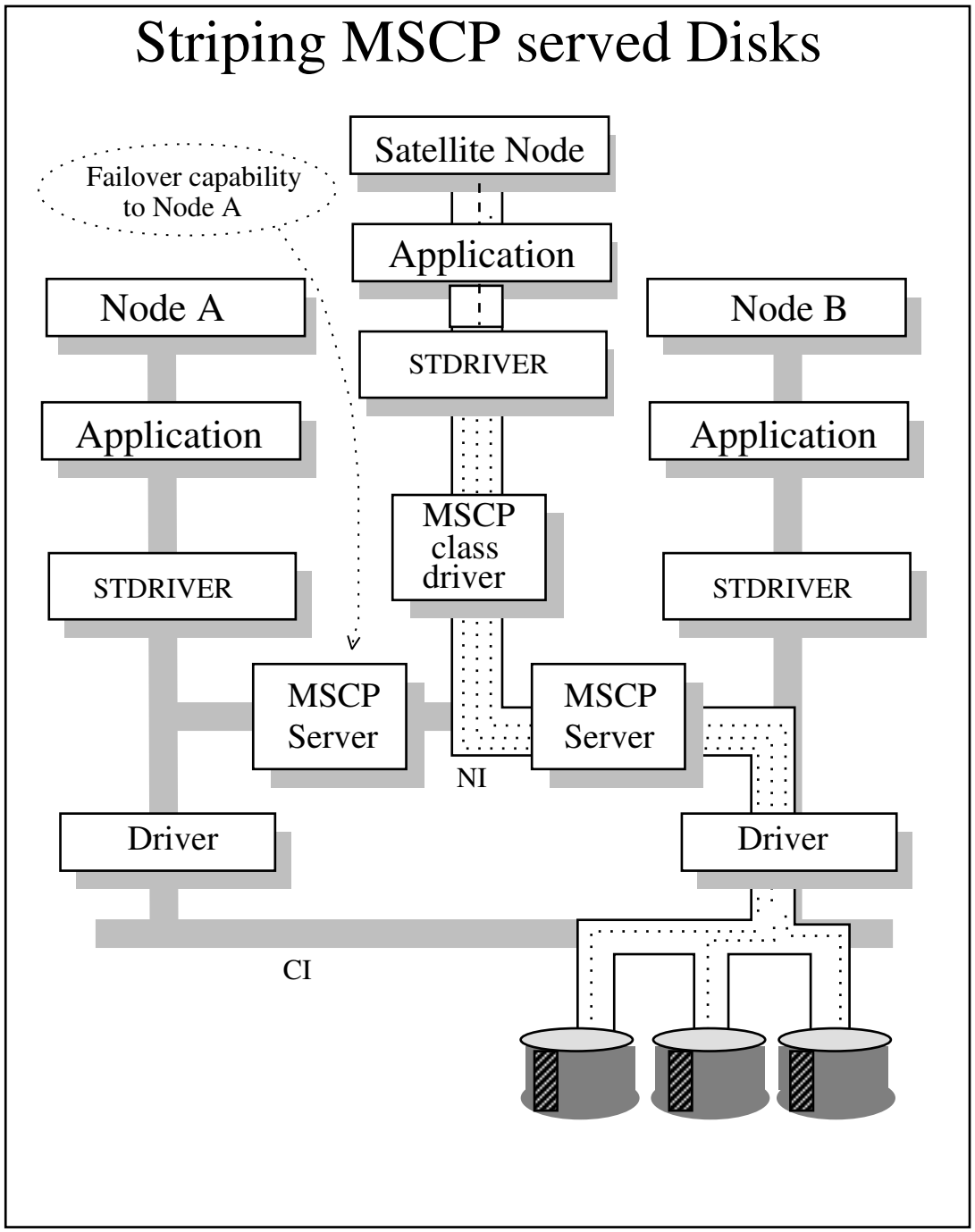
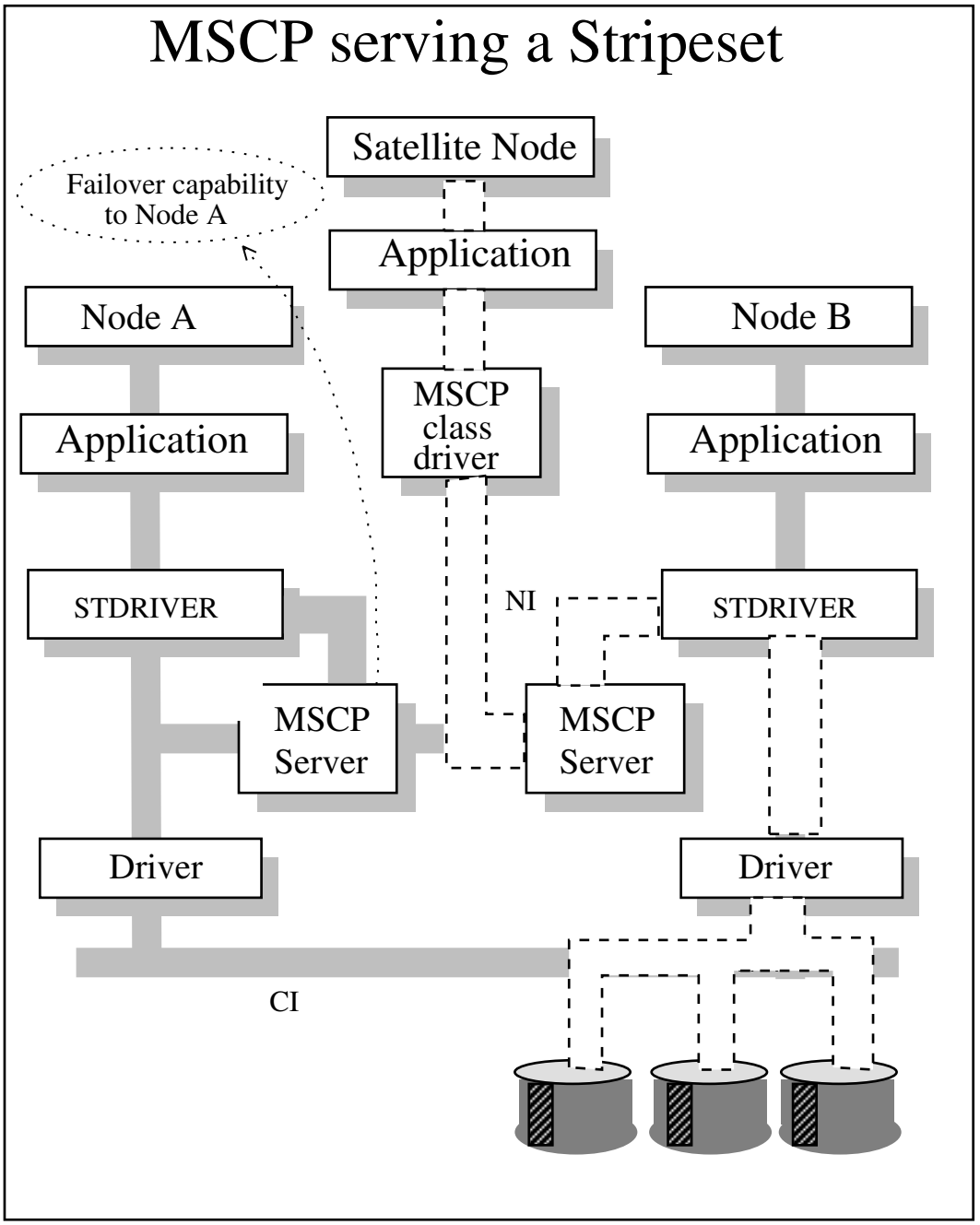


Figure 2–2 MSCP Serving the Stripeset Virtual Device on a VAXcluster System



2.2.2 Allocation Class

This section does not attempt to provide a full description of allocation classes and how to use them. It does explain how they impact disk striping.

2.2.2.1 What Is It and Why Is It Important?

An allocation class is a mechanism for ensuring that one disk, seen by multiple VAXcluster nodes, has one name within the VAXcluster system.

To MSCP serve a stripeset, the *stripeset* must have an allocation class equal to that of the serving node. To MSCP serve the stripeset members, the *members* must have an allocation class.

Plan and set your allocation class carefully, because changing an allocation class usually requires a system reboot.

2.2.2.2 How Do I Set the Allocation Class?

The allocation class of a **stripeset** is set by using:

- /ALLOCATION_CLASS qualifier on the STRIPE BIND command. This can be set to an integer value from 1 to 255. The first STRIPE BIND command sets the allocation class for all stripesets bound on that node *whether /ALLOCATION_CLASS is used or not.*

If the /ALLOCATION_CLASS qualifier is not used or does not equal that of the serving node, the stripeset virtual device cannot be served.

The method used to set the allocation class of a **member disk** depends on the type of controller that connects that member disk. Thus, if the member disk is controlled by:

- An HSC or ISE controller—the allocation class of that disk is set by an HSC or ISE parameter. For information on these parameters, refer to the *HSC Controller User Guide*.
- A Storage System Port (SSP) controller[1]—use the serving node's SYSGEN parameter, ALLOCLASS.

[1] SSP controllers are operated with a PUDRIVER and include UDA50, KDA50, KDB50, KDM70, KFQSA and RQDX3 controllers.

2.2.3 Enabling the MSCP Server

To serve either the stripeset or the stripeset members, you must load the MSCP server. You must then set either the stripeset or the stripeset members served. To do this:

- Use the SYSGEN utility[2] to ensure that
  - MSCP_LOAD =1, and
  - MSCP_SERVE_ALL =1 to serve ALL DEVICES,
    or
    MSCP_SERVE_ALL =2 to serve only LOCAL DEVICES.

Note: These are not dynamic parameters. If you change them, you must reboot the system before the changes become effective.

[2] See the *VMS System Generation Utility Manual* for further information on how to use the SYSGEN utility.

2.2.4 Creating a Stripeset on a VAXcluster System

Before you enter any of the commands in Table 2–6, consider the following:

1. The STRIPE BIND command (steps number 2 and 5 in the following table) must be entered *on each member* of a VAXcluster system that you want to have *local* access to the stripeset. This is also true for the MOUNT command (step number 6 in the following table).
2. If you plan to **MSCP serve** the:
   - **stripeset virtual device**–*specify an allocation class equal to the serving node* in each STRIPE BIND command in Table 2–6.
   - **stripeset members**–*do notspecify an allocation class* in each STRIPE BIND command in Table 2–6.

Note: If you have not enabled the MSCP server, see Section 2.2.3 before entering the following commands.

Table 2–6 Creating a Version 2.0 Stripeset on a VMS Version 5.4 (or later) VAXcluster System

Step	Node #	Command	Results
1	Node 1	*$ STRIPE INITIALIZE _$ DUA1,DUA2 TOM,DICK*	Initializes stripeset members; deletes all previous data on stripeset members.
2	Node 1	*$ STRIPE BIND DUA1,DUA2 _$ /DEVICE_NAME=STA1*	Creates stripeset virtual device named STA1 on Node 1.
3	Node 1	*$ INITIALIZE/CLUSTER_ SIZE=5 STA1 STRIPE_VOL*	Initializes a Files-11 file system on the stripeset and deletes all data on the stripeset virtual device.
4	Node 1	*$ MOUNT/SYSTEM STA1 _$ STRIPE_VOL*	Makes the Files-11 file system on the stripeset available on Node 1.
5	Node 2	*$ STRIPE BIND DUA1,DUA2 _$ /DEVICE_NAME=STA1*	Creates stripeset virtual device STA1 on Node 2. By using DEVICE_NAME=STA1 on Node 2, STA1 on Node 2 becomes the same device as STA1 on Node 1.
6	Node 2	*$ MOUNT/SYSTEM STA1 _$ STRIPE_VOL*	Makes the Files-11 file system on the stripeset available on Node 2. By using STA1 and STRIPE_ VOL on Node 2, the volume becomes the same as the STRIPE_VOLUME on Node 1.
7	Node 3	Repeat steps 5 and 6 on subsequent nodes.	

Steps 1 and 3 are entered once. Steps 2 and 4 are normally placed in the STRIPE$SYSTARTUP.COM file for each VAXcluster node that will have direct access to the stripeset members.

To ensure all binds occur prior to mount, avoid using the MOUNT/CLUSTER command. Locks synchronize the binding and mounting of the stripeset on various nodes of the VAXcluster system.

2.2.4.1 Striping on a VAXcluster System Prior to VMS Version 5.4 Software

Digital recommends you use VMS Version 5.4 software to implement striping in a VAXcluster environment. Failover does not occur on prior versions; thus, a stripeset becomes unavailable if the binding node fails.

Prior to VMS Version 5.4 software, you must bind the stripeset on one node and MSCP serve the stripeset virtual device to all other VAXcluster nodes which require access to the stripeset.

Before you create this stripeset:

1. Choose the VAXcluster node that will MSCP serve the stripeset.
2. Enable the MSCP server on the VAXcluster node you have chosen to serve the stripeset by:
   - Using SYSGEN to ensure that MSCP_LOAD=1 and MSCP_SERVE_ALL=1 or 2 [1] See Section 2.2.3 for more information on how to do this.

     NOTE: These are not dynamic parameters. If you change them, you must reboot the system before they become effective.
3. Follow the instructions in Table 2–7.

Table 2–7 Creating a Stripeset on a VMS Version 5.3 (or earlier) VAXcluster with VAX Disk Striping Driver Version 2.0 Software

Step	Node #	Command	Results
1	Node 1	*$ STRIPE INITIALIZE _$ DUA1,DUA2 TOM,DICK*	Initializes stripeset members; deletes all previous data on stripeset members.
2	Node 1	*$ STRIPE BIND DUA1,DUA2 _$ /DEVICE_ NAME=STA1/ALLOCATION_ CLASS=2*	Creates a stripeset virtual device named STA1, with an allocation class of 2, on Node 1.
3	Node 1	*$ INITIALIZE/CLUSTER_SIZE=5 2STA1 STRIPE_VOL*	Initializes a Files-11 file system on the stripeset and deletes all data on the stripeset virtual device.
4	Node 1	*$ MOUNT/SYSTEM 2STA1 _$ STRIPE_VOL*	Makes the Files-11 file system on the stripeset available on Node 1.
5	Node 2	*$ MOUNT/SYSTEM 2STA1 _$ STRIPE_VOL*	Makes the Files-11 file system on the stripeset available on Node 2. By using 2STA1 and STRIPE_VOL on Node 2, the volume becomes the same as the STRIPE_VOLUME on Node 1.
6	Node 3	Repeat step 5 on subsequent nodes.	

[1] Setting the value to 1 enables MSCP serving to ALL DEVICES; setting the value to 2 enables MSCP serving to LOCAL DEVICES.

Example 2–2 Example—Creating a Stripeset in a VAXcluster Environment

```
              ********************************************************
                        Commands entered from Node 1
              ********************************************************

NODE1> STRIPE INIT $1$DUA210, $1$DUA211 STRP1, STRP2

%STRIPE-I-VOLID, unit _$1$DUA210: contains volume IVP$_TOPS10_
%STRIPE-I-VOLID, unit _$1$DUA211: contains volume IVP$_TOPS20_
Proceed to initialize specified units? [Y/N] (N) : Y
%STRIPE-I-INIT, initializing member unit $1$DUA210 as volume STRP1
%STRIPE-I-INIT, initializing member unit $1$DUA211 as volume STRP2
%STRIPE-I-MOUNTING, mounting volume STRP1 on unit $1$DUA210
%MOUNT-I-MOUNTED, STRP1        mounted on _$1$DUA210: (NODE1)
%STRIPE-I-MOUNTING, mounting volume STRP2 on unit $1$DUA211
%MOUNT-I-MOUNTED, STRP2        mounted on _$1$DUA211: (READY)

NODE1> STRIPE BIND $1$DUA210, $1$DUA211/DEVICE_NAME=STA1
%STRIPE-S-BOUND, stripe set STA1 defined
%STRIPE-I-CLUSTER_SUPPORT, DEVICE  CLUSTER WIDE AFTER BINDING ON EACH NODE
%STRIPE-I-WSYNDEQ, Waiting for synch-lock to dequeue

NODE1> SHOW DEV STA1

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
STA1:                   Online               0


NODE1> INITIALIZE/CLUSTER_FACTOR=5 STA1 STRIPE$1
NODE1> MOUNT/SYSTEM STA1 STRIPE$1
%MOUNT-I-MOUNTED, STRIPE$1     mounted on _STA1:

NODE1> SHOW DEV STA1

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
STA1:                   Mounted              0  STRIPE$1       4747790     1   1

NODE1>

              ********************************************************
                        Commands entered from Node 2
              ********************************************************

NODE2> STRIPE BIND $1$DUA210, $1$DUA211/DEVICE_NAME=STA1
%STRIPE-I-MOUNTING, mounting volume STRP2 on unit $1$DUA211
%MOUNT-I-MOUNTED, STRP2        mounted on _$1$DUA211: (NODE2)
%STRIPE-S-BOUND, stripe set STA1 defined
%STRIPE-I-CLUSTER_SUPPORT, DEVICE  CLUSTER WIDE AFTER BINDING ON EACH NODE
%STRIPE-I-WSYNDEQ, Waiting for synch-lock to dequeue

NODE2> MOUNT/SYSTEM STA1 STRIPE$1
%MOUNT-I-MOUNTED, STRIPE$1     mounted on _STA1:
NODE2> SHOW DEV STA1

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
STA1:                   Mounted              0  STRIPE$1       4747790     1   2

NODE2>
```

2.3 Using Shadowed Disks

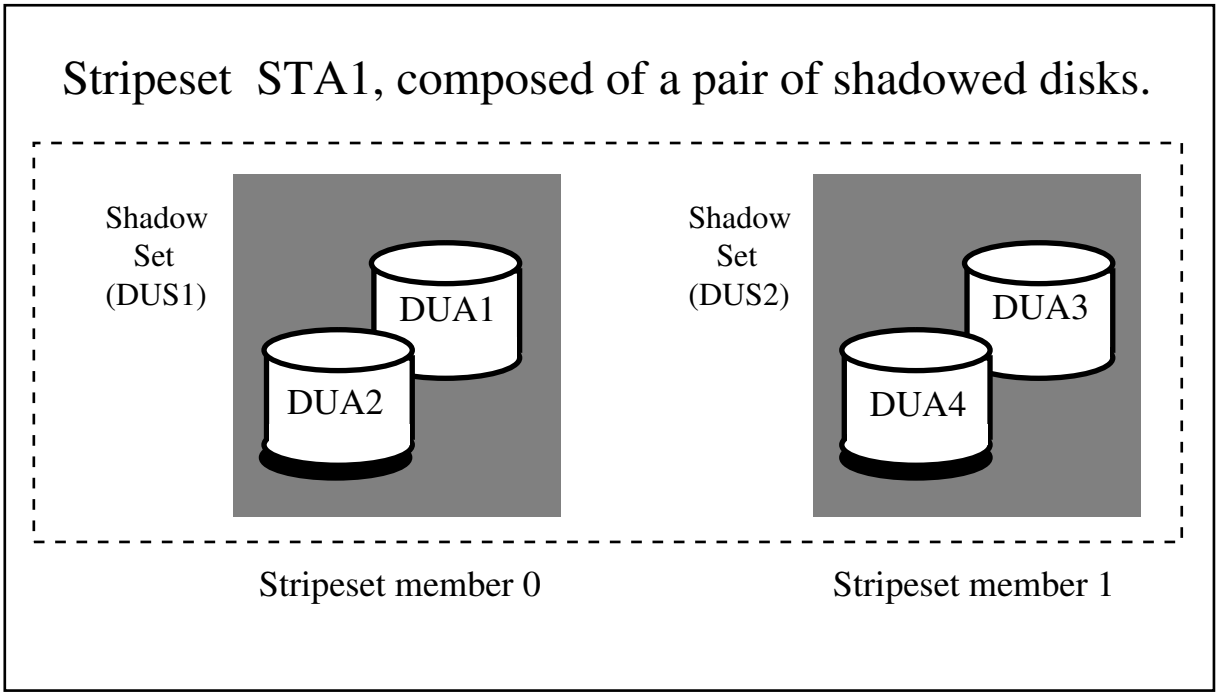
This section does not attempt to provide a full description of volume shadowing. For additional information, refer to the *VAX Volume Shadowing Manual*.

Note: Digital does not recommend splitting your striped shadowsets and creating duplicate stripesets in order to do backups. For more information on this, see Section 2.6.2, Maintaining a Stripeset.

Controller-based Volume Shadowing (Phase I) can be used with disk drives connected to HSC controllers. Host-based Volume Shadowing (Phase II) requires VMS Version 5.4 or later, and can be used with all Digital Standard Architecture (DSA) devices.

One does **not** shadow stripesets; one stripes shadow sets. Furthermore, *each* stripeset member unit must be shadowed *individually*. Figure 2–3, Striping Shadow Sets, shows a stripeset that is composed of two member units; each is a shadowed set of two disks.

Figure 2–3 Striping Shadow Sets



Use the following series of commands to implement controller-based[1] shadowing on the stripeset members, shown in Figure 2–3. **All four devices should be dismounted and on line before you execute the sequence of commands.**

[1] If you want to use host-based shadowing, replace DUS1 and DUS2 with DSA1 and DSA2.

```
 1  $ STRIPE INITIALIZE DUA1,DUA3 MEM0,MEM1
 2  $ DISMOUNT/NOUNLOAD DUA1
 3  $ DISMOUNT/NOUNLOAD DUA3
 4  $ MOUNT/SYSTEM DUS1/SHADOW=(DUA1) MEM0
 5  $ MOUNT/SYSTEM DUS2/SHADOW=(DUA3) MEM1
 6  $ STRIPE BIND DUS1,DUS2 /DEVICE_NAME=STA1
 7  $ INIT STA1 STRIPE_VOL
 8  $ MOUNT STA1 STRIPE_VOL
 9  $ MOUNT/SYSTEM DUS1/SHADOW=(DUA2) MEM0
10  $ MOUNT/SYSTEM DUS2/SHADOW=(DUA4) MEM1
```

The shadowed stripeset STA1 is now available for stripeset file operations.

2.3.1 Combining Disk Striping with Volume Shadowing in a VAXcluster System

Disk striping can be used with either controller-based (Phase I) or host-based (Phase II) Volume Shadowing in a VAXcluster environment. As with single CPU systems, the stripeset is created from members that are, themselves, shadow sets.

In the following example, both CPUs of a two-node VAXcluster system have direct access to all disks that comprise the two shadow sets; both disk striping and volume shadowing have been activated. Each node will create two shadow set virtual disks. A stripeset virtual device (comprised of the two shadow set virtual disks) will be created on each node.

Note: Steps 1 and 7 should only be entered once, as they destroy all existing data on the disks.

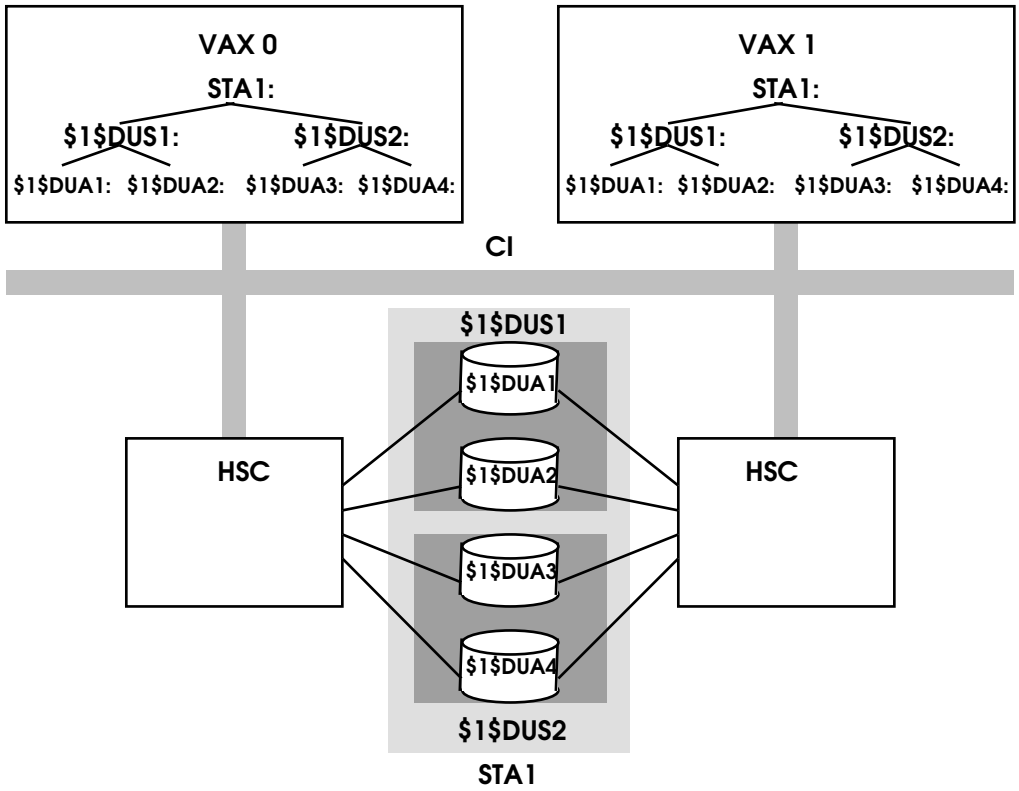
Use the following series of commands to build this stripeset.

```
 1  VAX_0$ STRIPE INITIALIZE $1$DUA1,$1$DUA3 MEM0,MEM1
 2  VAX_0$ DISMOUNT/NOUNLOAD $1$DUA1
 3  VAX_0$ DISMOUNT/NOUNLOAD $1$DUA3
 4  VAX_0$ MOUNT/SYSTEM $1$DUS1/SHADOW=($1$DUA1) MEM0
 5  VAX_0$ MOUNT/SYSTEM $1$DUS2/SHADOW=($1$DUA3) MEM1
 6  VAX_0$ STRIPE BIND $1$DUS1,$1$DUS2/DEVICE_NAME=STA1
 7  VAX_0$ INITIALIZE STA1 STRIPE_VOL
 8  VAX_0$ MOUNT/SYSTEM STA1 STRIPE_VOL
 9  VAX_0$ MOUNT/SYSTEM $1$DUS1/SHADOW=($1$DUA2) MEM0
10  VAX_0$ MOUNT/SYSTEM $1$DUS2/SHADOW=($1$DUA4) MEM1
11  VAX_1$ MOUNT/SYSTEM $1$DUS1/SHADOW=($1$DUA1,$1$DUA2) MEM1
12  VAX_1$ MOUNT/SYSTEM $1$DUS2/SHADOW=($1$DUA3,$1$DUA4) MEM0
13  VAX_1$ STRIPE BIND $1$DUS1,$1$DUS2/DEVICE_NAME=STA1
14  VAX_1$ MOUNT/SYSTEM STA1 STRIPE_VOL
```

When binding and mounting an existing stripeset of shadow sets, steps 11 through 14 **must** be executed on each VAXcluster CPU node from which you want access to the stripesets. These steps are usually placed in the STRIPE$SYSTARTUP.COM procedure.

Also, each disk and shadow set virtual disk name includes an allocation class. This is necessary to allow the members to fail over when an HSC fails.

Figure 2–4 Striping Shadow Sets in a VAXcluster Environment



When a VAXcluster node does not have direct access to the physical disks comprising a stripeset made of shadow sets, the stripeset should be MSCP served to that node. Or, you may serve the physical disks to the node. (Digital does not recommend you serve the shadow set virtual disks to the node. Host-based shadow set virtual disks cannot be MSCP served.) See Section 2.2.3, Enabling the MSCP Server, and Section 2.2.2, Allocation Class.

2.3.2 Data Protection Using VAXsimPLUS Software

VAXsimPLUS software attempts to predict physical disk [1] failures, and will notify the operator in such an event.

If the suspect disk is a single-member shadow set, VAXsimPLUS software can add a spare disk to that shadow set. As a result, all data will be copied from the suspect disk to the spare disk, thus preventing the suspect disk's failure from destroying unique data. In this way, VAXsimPLUS will improve data reliability of your stripeset members.

Stripesets not made of shadow sets can still have VAXsimPLUS software available to notify the operator that a DSA disk is about to fail.

2.4 Unbinding the Stripeset

When you no longer need a stripeset, you can use the STRIPE UNBIND command shown in Table 2–8.

Table 2–8 STRIPE UNBIND Command Example and Results

Command	Results
`$ DISMOUNTDISK$STRIPE`	• Dismounts the volume with logical name "DISK$STRIPE."
`$ STRIPE UNBIND DISK$STRIPE`	• Places the stripeset virtual device off line.

The STRIPE UNBIND command does **not** affect the data stored on the member volumes, but produces these results in the following order:

1. frees the stripeset member units (after all pending I/O requests have been processed),
2. dismounts member volumes, and
3. deassigns the logical name assigned when the stripeset was bound.

Whether it has been MSCP served or not, the stripeset virtual device does not disappear; it is set off line and is unusable until a STRIPE BIND command is reentered with same unit number.

[1] At this writing, VAXsimPLUS only supports Digital's DSA disks. Please consult the VAXsimPLUS documentation to see what is currently supported.

2.5 Stripeset Configuration Restrictions

A stripeset may NOT be used as a:

- bootable system disk,
- a disk to hold the system dump file, or
- a VAXcluster quorum disk.

In each case, low-level operating system software performs these functions and that software is unaware of the VAX Disk Striping Driver software.

2.6 Managing Stripesets Larger Than 8.5 GB

Currently, the VMS Files-11 system software does not support disks larger than 8.5 GB. Because it is sometimes desirable to build stripesets that are larger than 8.5 GB, VAX Disk Striping Driver software includes an optional feature which limits the size of the stripeset virtual disk to 8.5 GB. This feature is controlled by the /FILES_11 qualifier to the STRIPE INITIALIZE command.

STRIPE INITIALIZE/FILES_11 limits the size of the stripeset virtual device to 8.5 GB. Because the consequences of using a Files-11 file system on a disk larger than 8.5 GB can be great, **the /FILES_11 qualifier is the default**.

When using a stripeset with a file system that is capable of using disks larger than 8.5 GB, add the /NOFILES_11 qualifier to the STRIPE INITIALIZE command used to create the stripeset.

2.6.1 Determining the Stripeset's Maximum Transfer Size

Every VMS disk device has a maximum transfer size, and a stripeset virtual disk is no exception. When an I/O request is larger than the disk's maximum transfer size, the SYS$QIO system service routines will divide the request into segments and execute each of these segments as if it were a separate I/O request. Because the SYS$QIO system service routines serialize the execution of the segmented I/O requests, poor performance may result.

The stripeset virtual disk's maximum unsegmented transfer size is displayed by the STRIPE LIST command. It is a function of

- maximum transfer size of the stripeset member disks, and
- the /TRANSFER_BLOCK qualifier, specified when the stripeset is bound.

The maximum transfer size of the stripeset member disks may be limited by the disk's controller, the adapter that connects the controller to the host, and VMS software such as the MSCP server and the host-based volume shadowing driver. Cases where the stripeset member's maximum transfer size may be a problem include:

- When Storage System Port (SSP) disk controllers limit the maximum transfer size to 127 blocks. SSP disk controllers are all those which use the PUDRIVER, including the UDA50, KDA50, KDB50, KDM70, KFQSA and RQDX3 controllers.
- When host-based (Phase II) volume shadowing is used to shadow a stripeset member, the stripeset member maximum transfer size is 127 blocks.

The maximum transfer size of a stripeset virtual disk is approximately the number of members multiplied by the maximum transfer size for the members. More specifically, it is the number of whole chunks that fit inside the member's maximum transfer size multiplied by the stripeset's chunk size, multiplied by the number of members. Consider a stripeset of three RA70 disk drives on a KDM70 controller, and assume that the stripeset chunk size is the default, 31 logical blocks. The KDM70 controller has a maximum transfer size of 127 blocks, so four chunks fit within one maximum size transfer. That means that the stripeset's maximum transfer size will be:

```
    4 whole-chunks within the maximum transfer size of a member
   31 logical blocks per chunk
x   3 members per stripeset
----
  372 logical blocks
```

Large transfers may require many system page table entries and pages of nonpaged pool. When a stripeset is bound, the /TRANSFER_BLOCK qualifier allows the system manager to limit the consumption of system page table entries by limiting the maximum transfer size of the stripeset virtual disk.

If in the above example, the stripeset were bound with the /TRANSFER_BLOCK =496, then the maximum transfer size of the stripeset virtual disk would be limited by the members to 372. However, if the stripeset were bound with the default /TRANSFER_BLOCK =127, the stripeset virtual disk would be limited to a 127 block transfer size.

When specifying a larger value for /TRANSFER_BLOCK, it may be necessary to increase the number of system page table entries available to the VMS software. This is done by increasing the SYSGEN parameter SPTREQ and rebooting the VMS software. See Section 3.3, Post-Installation, for further information.

2.6.2 Maintaining a Stripeset

As with any VMS disk, it is occasionally necessary to verify, backup, and defragment a stripeset. Utilities designed to verify, backup, and defragment a disk should be used on the stripeset virtual disk.

It is important to understand that these utilities should NOT be used on the individual members of the stripeset, but on the *virtual disk itself*. **Using such utilities on the individual stripeset members can result in lost or corrupted data.**

Note: Digital does not recommend splitting your striped shadowsets and creating duplicate stripesets in order to do backups.

Customers who find it necessary to follow this procedure and who are using shadowset members that are no longer in a shadowset must mount the shadowset members with the following commands:

```
$ MOUNT/NOSYSTEM/OVER=SHADOW_MEMBERSHIP $1$DUAxx
$ MOUNT/NOSYSTEM/OVER=SHADOW_MEMBERSHIP $1$DUAyy
```

Then enter the STRIPE BIND command:

```
$ STRIPE BIND/NOSYSTEM $1$DUAxx, $1$DUAyy
$_  /DEVICE_NAME=STAzz
```

STAzz must not be on line prior to entering the STRIPE BIND command on the VAXcluster nodes.

3 Installation Guide

After you receive the VAX Disk Striping Driver software in the form of a standard VMS software installation kit, perform the following steps.

1. Review Section 3.1.1, Configuration Planning Checklist.
2. Perform the preinstallation procedures using the Section 3.1.5, Preinstallation Procedure.
3. Install the STDRIVER software using the VMSINSTAL utility.
4. Perform the post-installation procedures.

3.1 Preinstallation Planning

In this chapter, you will find the following information.

1. Configuration Planning Checklist
2. Preinstallation Requirements
3. Preinstallation Restrictions
4. Time Needed for Installation
5. Preinstallation Procedure
6. Adjusting SYSGEN Parameters
7. Upgrading Disk Striping Software From Version 1.0 to Version 2.0
8. Registering the License Product Authorization Key

3.1.1 Configuration Planning Checklist

Following is a checklist of topics to consider before you install the VAX Disk Striping software. **Refer to Chapter 2 for configuration planning details.** (Refer to the *VAX Disk Striping Driver Guide to Performance and Tuning* for more specific performance information.)

1. Review your site's data transfer rates, I/O size, and queue length. (See Section 1.2.1 and Section 1.2.3.)
2. Determine appropriate configuration to maximize
   - stripeset access from each node,
   - load balancing,
   - high throughput, and
   - data reliability for your application. (See Section 1.2.2 and Section 1.2.3.)
3. Review Section 3.1.7.

4 Plan to preserve data on potential stripeset members.

3.1.2 Preinstallation Requirements

To install the VAX Disk Striping Driver software, you must:

- be running VMS Version 5.0 or later,
- have 2500 free blocks on the system disk during installation, and
- allow 1500 blocks of storage for striping software, after installation is complete.

3.1.3 Preinstallation Restrictions

A stripeset may not be used as a:

- bootable system disk,
- VAXcluster quorum disk, or
- a disk for the system dump file.

Stripesets using VMS Files-11 file system cannot be larger than 8.5 GB. See Section 2.6 for a complete discussion of stripeset size limitations and the STRIPE/INITIALIZE qualifiers required.

3.1.4 Time Needed for Installation

The time needed for installation and verification depends on the system configuration and the load on the system. The sample installation took 8 minutes; however, time required to build a stripeset may take considerably longer due to:

- hardware configuration adjustments, and
- VMS SYSGEN parameter adjustments that require a system reboot.

3.1.5 Preinstallation Procedure

Perform the following tasks before installing the VAX Disk Striping Driver software.

Table 3–1 Preinstallation Procedures

Task	Step	Action
Save data	1	Log into the SYSTEM account.
	2	Perform an image backup to preserve data on potential stripeset members.

Table 3–1 (Cont.) Preinstallation Procedures

Task	Step	Action
Adjust SYSGEN parameters	3	Examine SYSGEN parameters; adjust, if necessary. (See Section 3.1.6.)
		IF . . . / **THEN . . .**
		you need to adjust SYSGEN parameters / you must reboot the system before using the stripeset. (If you are performing step 4, you may want to delay rebooting until after its completion.)
Prepare for Version 1.0 upgrade	4	
		IF . . . / **THEN . . .**
		you are upgrading from Version 1.0 to Version 2.0 / perform the steps in 3.1.7, then go to step 5 in this table. **Note: You must reboot the system** before using the stripeset to replace the Version 1.0 driver with the Version 2.0 driver.
		you are not upgrading from Version 1.0 / skip this step and go to step 5.
Register the PAK	5	Register the license Product Authorization Key (PAK). (See Section 3.1.8.)
Install Kit	6	Go to Section 3.2 to install the VAX Disk Striping Driver software kit.

3.1.6 Adjusting SYSGEN Parameters

It is unlikely that you will need to modify the SYSGEN parameters unless your applications issue very large I/O requests to the stripeset virtual disk.

Certain application characteristics, such as the number of stripesets and the maximum data transfer sizes, may result in a strain on the system's resources. The striping driver may consume enough system page table (SPT) entries and/or nonpaged pool (NPP) memory to require modifying SYSGEN parameters controlling these resources.

SPT entries are allocated by each STRIPE BIND command and deallocated by each STRIPE UNBIND command. The number of SPT entries used by a stripeset is:

```
4 + (2 * B)
```

where B is the maximum transfer size for the stripeset virtual device. See Section 2.6.1, Determining the Stripeset's Maximum Transfer Size, to determine the maximum transfer size of a stripeset.

Unless there are sufficient SPT entries, the STRIPE BIND command will fail with status SS$_INSFSPTS. If this occurs, use AUTOGEN to increase the SYSGEN parameter SPTREQ by 4 + (2 * B) for each stripeset.

As stripeset virtual disks are used, memory in nonpaged pool is temporarily allocated. Specifically, memory is allocated when a transfer larger than a trip is issued to a stripeset, and that memory is deallocated when the transfer completes. The following table describes the memory required for each buffer page for active I/O requests to the stripeset.

Dynamic System Resources Consumed by Striping Driver Transfers
(Numbers in the table are per buffer page of I/O outstanding)

Byte Count < Trip	Byte Count > Trip and Buffer is Page-aligned	Byte Count > Trip and Buffer NOT Page-aligned
0 Bytes of NPP*	4 Bytes of NPP	4 Bytes of NPP, and 1 Page of NPP

* NPP – nonpaged pool

3.1.6.1 Detecting SYSGEN Problems

If you suspect a problem with the amount of nonpaged pool available, use MONITOR POOL to determine the minimum amount of nonpaged pool available. If there is a shortage, use AUTOGEN to increase the SYSGEN parameter NPAGEDYN.

The following example shows how to calculate the system resources used for striping.

Suppose your stripeset has a maximum transfer size of 1000 pages and a trip size of 500 pages. That means binding the stripeset will consume 2004 SPT entries.

Now, suppose your applications keep two 1000 page I/O requests outstanding at all times (double buffering) and the application's buffer is not page aligned.

This means the application's stripeset I/O will consume:

```
      516  bytes of NPP per outstanding unaligned transfer > 1 trip
    x   2  I/O requests outstanding
   x 1000  pages per I/O request
   ------
1,032,000     bytes of nonpaged pool.
```

If the MONITOR POOL command shows less than 1032 minimum dynamic Kbytes available (1,032,000 bytes) of nonpaged pool then you should increase NPAGEDYN.

3.1.6.2 Using AUTOGEN

If you need to modify SYSGEN parameters, Digital recommends you use the AUTOGEN utility because it automatically readjusts all related parameters. Refer to the*VMS Guide to Performance Management* in the VMS documentation set for detailed instructions for using AUTOGEN to set system parameters.

Briefly, you must first edit the SYS$SYSTEM:MODPARAMS.DAT file to specify the parameter increments you require. Nondynamic SYSGEN parameter changes require a reboot to take effect.

To readjust all related system parameters and reboot the system, enter the following command:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT
```

Rebooting the system enables the new parameter settings.

3.1.7 Upgrading Disk Striping Software From Version 1.0 to Version 2.0

Before installing the VAX Disk Striping Driver (STDRIVER) Version 2.0 software:

1. Perform steps 1 through 4 in Section 3.1.5.
2. Do an image backup [1] of your Version 1.0 stripeset[2].
3. Dismount the Version 1.0 stripeset virtual device.
4. Unbind the Version 1.0 stripeset with the STRIPE UNBIND command.
5. Disable system startup procedures for Version 1.0 to prevent:
   - binding or mounting, during any reboot, prior to completing all Version 2.0 post-installation activities,
   - binding the Version 2.0 stripesets with the wrong allocation class,
   - the Version 1.0 STRIPE BIND and STRIPE INITIALIZE commands from being executed, as they have changed in Version 2.0.
6. Reboot the system to unload Version 1.0 STDRIVER.
7. Resume Section 3.1.5 at step 5.

3.1.8 Registering the License Product Authorization Key

Before you can install STDRIVER, you must register the PAK. To do this, execute the interactive command procedure, VMSLICENSE.COM, which provides you with a *question and answer* interface that simplifies data entry and registration.

Enter the following command and press Return.

```
@SYS$UPDATE:VMSLICENSE
```

VMSLICENSE displays a menu of options; enter 1 and press Return. The following message is displayed:

```
Do you have your Product Authorization Key?
```

[1] See the *VMS Backup Manual*.

[2] The size of the Version 2.0 stripeset may differ from the Version 1.0 stripeset size. To ensure the RESTORE command succeeds, Digital recommends at least one percent of the blocks on the Version 1.0 stripeset be free.

Make sure that you have the PAK for the VAX Disk Striping Driver for VMS. Enter Y and press Return .

Answer the questions using the data from the PAK. Be sure to enter each field **exactly** as it appears on the PAK. (Do not enter any fields that are blank on the PAK.)

For additional information on how to manage software licenses, refer to:

- *A Practical Guide to Managing Software Licenses*
- *VMS License Management Utility Manual*

3.2 Installing STDRIVER

The STDRIVER installation procedure automatically updates the DCL command tables to include the STRIPE command.

To install STDRIVER:

1. Log into the SYSTEM account,
2. Enter @SYS$UPDATE:VMSINSTAL Return , and
3. Fill in the appropriate information for your system at each VMSINSTAL prompt.

Example 3–1 is a sample installation.

Example 3–1 Sample Installation

```
$ @SYS$UPDATE:VMSINSTAL
 VAX/VMS Software Product Installation Procedure V5.4

It is 1-FEB-1991 at 09:18.

Enter a question mark (?) at any time for help.

Are you satisfied with the backup of your system disk [YES]? YES Return
Where will the distribution volumes be mounted: MUA0: Return

Enter the products to be processed from the first distribution volume set.
Products: stripe020 Return
Enter installation options you wish to use (none): Return
The following products will be processed:
  STRIPE V2.0

 Beginning installation of STRIPE V2.0 at 09:51

%VMSINSTAL-I-RESTORE, Restoring product save set A ...
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS$HELP.
Do you want to purge files replaced by this installation [YES]? YES Return

 Product:      STRIPING
 Producer:     DEC
        Version:      02.0
        Release Date: 01-JUNE-1991

Does this product have an authorization key registered and loaded? YES Return

 The stripe driver being built will allow for  local
 access to stripesets by nodes that have local access to the
 stripeset members.
```

Example 3–1 Cont'd on next page

Example 3–1 (Cont.) Sample Installation

```
%STRIPE-I-VMSVER, Assembling STDRIVER for VMS version 5

To add the striping driver startup procedure to the system startup procedure,
place the following command in your STARTUP.COM file.

 @SYS$STARTUP:STRIPE$STARTUP.COM

Be sure to read the documentation concerning STRIPE$SYSTARTUP.COM in the
VAX Disk Striping Driver for VMS Guide to Operations.

 The primary installation phase has completed.  Be sure to read the
 release notes, and note especially that you may need to adjust the
 SYSBOOT (SYSGEN) parameters which control the amount of nonpaged pool,
 and number of SPTs for STRIPE to work.

        A test package to verify that the Striping Driver works correctly
 has been placed on SYS$TEST: and may be run with the following
 command:

$ @SYS$TEST:STRIPE$TEST

            WARNING!!  WARNING!! WARNING!! WARNING!!

       This test requires the use of two disks.
       Data on the disks used in this test will be destroyed!!!!!

      Any V1.0 stripe sets will need to be backed up, stripe initialized,
      stripe bound, VMS initialized, mounted, and restored to take advantage
      of the new features.
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
 Installation of STRIPE V2.0 completed at 09:54

Enter the products to be processed from the next distribution volume set.
Products: Return
 VMSINSTAL procedure done at 09:55
$
```

3.3 Post-Installation

After you install the STDRIVER on your system, you must make some adjustments. The following sections provide information for verifying the installation of STDRIVER and modifying your files to allow for disk striping on your system.

3.3.1 After Installing Disk Striping Software

According to your new stripeset configuration and after installation of the VAX Disk Striping Driver software Version 2.0, complete the following tasks:

1. Initialize the Version 2.0 stripeset with the STRIPE INITIALIZE command.
2. Bind the Version 2.0 stripeset with the STRIPE BIND command.
3. Initialize the Version 2.0 stripeset virtual device using the VMS INITIALIZE command.

4 See the following table:

IF . . .	THEN . . .
you have previously done an image backup and need to restore that data to the Version 2.0 stripeset virtual device	follow these steps: • mount the Version 2.0 stripeset virtual device as a foreign volume, • use the BACKUP utility to restore any data targeted for the Version 2.0 stripeset, and • dismount the Version 2.0 stripeset virtual device.

5 Mount the V2.0 stripeset virtual device according to the planned configuration for your stripeset.

6 Enter the @SYS$STARTUP:STRIPE$STARTUP.COM command. (See Section 3.3.2.)

7 See the following table:

IF . . .	THEN . . .
you want to implement Version 2.0 striping on other nodes of your VAXcluster	enter these commands: • INSTALL/REPLACE SYS$COMMON:[SYSLIB]DCLTABLES.EXE/OPEN/SHARE/LINK • INSTALL/REPLACE SYS$COMMON:[SYSMSG]STRIPEMSGFIL.EXE/OPEN/SHARE

3.3.2 Startup File Modifications

To automatically start STDRIVER after a system bootstrap, include the following command in your SYSTARTUP_V5.COM file:

@SYS$STARTUP:STRIPE$STARTUP.COM

In addition to starting STDRIVER, all stripesets have to be rebound. This is accomplished by the STRIPE$SYSTARTUP.COM file. The STDRIVER installation procedure includes the following null STRIPE$SYSTARTUP.COM file template:

```
$!Template for binding stripe sets during system startup. Each stripe set must
$!be bound when the system is rebooted.  This file contains the commands to
$!perform this activity.  If this file is present it is run by the
$!STRIPE$STARTUP.COM which must be included in the system startup procedure.
$!The STRIPE BIND command binds units that have been intialized as a stripe
$!set into pseudo_device.  A system wide logical name can be associated with
$!the stripe set pseudo_device. The following is an example of a STRIPE BIND
$!command:
$!       $ STRIPE BIND DUA1,DUA2 DISK$STRIPE/DEVICE_NAME = STA0
$!
$!       $ MOUNT/SYSTEM DISK$STRIPE Pagefile
```

Edit this file to include STRIPE BIND and MOUNT commands for all stripesets used on this system.

System initialization time may be reduced by submitting multiple batch jobs, each of which contain the STRIPE BIND and MOUNT commands for the stripeset. This allows the stripesets to be bound in parallel.

3.4 Files Added to System

A number of files are involved in the installation of STDRIVER. Some of these files are used in the installation procedure and then deleted; others are stored permanently in various directories of the system.

3.4.1 Temporary Files

The following files are used and then deleted from the system:

```
KITINSTAL.COM              STRIPE.OLB
STDRV.MAR                  STRIPENOLCK.OLB
STDRV_SYM.MAR              STDRIVER.OBJ
STRIPE$HELP.HLP            STDRV_UCB.OBJ
STRIPE$IVP.TEMPLATE        STDRV_LCK.OBJ
STRIPE_SYM.MAR             STDRIVERNOLCK.OBJ
STRIPEDEF.MAR              STDRV_UCB_NOLCK.OBJ
STRIPE.CLD                 STDRV_LCK_NOLCK.OBJ
```

3.4.2 Permanent Files

The following files are modified and remain in the specified directories:

```
SYS$HELP:HELPLIB.HLB
SYS$LIBRARY:DCLTABLES.EXE
```

The following files are added to the specified directories:

```
SYS$HELP:STRIPE020.RELEASE_NOTES
SYS$LOADABLE_IMAGES:STDRIVER.EXE
SYS$MANAGER:STRIPE$DEINSTALL.COM
SYS$MESSAGE:STRIPEMSGFIL.EXE
SYS$STARTUP:STRIPE$STARTUP.COM
SYS$STARTUP:STRIPE$SYSTARTUP.COM
SYS$SYSTEM:STRIPE.EXE
SYS$SYSTEM:STRIPE$ERRLOG.EXE
SYS$TEST:STRIPE$TEST.COM
```

3.5 Verifying the Installation

To verify the installation, log into the SYSTEM account and execute the command procedure in SYS$TEST:STRIPE$TEST.COM. Example 3–2 is a typical verification session.

Example 3–2 Sample Verification

```
@SYS$TEST:STRIPE$TEST


Beginning the STRIPE  IFT V2.0 Installation Verification Procedure.

 The STRIPE installation verification procedure requires the use
 of 2 disks that can be initialized and bound into a stripe set.

 Neither disk may be mounted or allocated.

    WARNING
  THE DATA ON BOTH OF THESE DISKS WILL BE LOST.

Do you wish to proceed with the installation verification? [NO]: YES

Enter disk #1: $1$DUA210

Enter disk #2: $1$DUA211

Do you wish to enter another disk? [YES]: YES

  Initializing test stripeset...

 If you already have stripesets created use the same allocation
 class for this test. Other wise choose the allocation class you
 will be using for your system.
  ... An allocation class is not required...
 * See the users guide for more information on allocation classes *

Please input an allocation class or <CR> for none: Return

%STRIPE-I-VOLID, unit _$1$DUA210: contains volume STRP1
%STRIPE-I-VOLID, unit _$1$DUA211: contains volume STRP2

Proceed to initialize specified units? [Y/N] (N) :  YES
%STRIPE-I-INIT, initializing member unit DUA210 as volume IVP$_TOPS10_
%STRIPE-I-INIT, initializing member unit $1$DUA211 as volume IVP$_TOPS20_
%STRIPE-I-MOUNTING, mounting volume IVP$_TOPS10_ on unit DUA210
%MOUNT-I-MOUNTED, IVP$_TOPS10_ mounted on _$1$DUA210: (AIM1)
%STRIPE-I-MOUNTING, mounting volume IVP$_TOPS20_ on unit $1$DUA211
%MOUNT-I-MOUNTED, IVP$_TOPS20_ mounted on _$1$DUA211: (READY)
  Creating stripe device...
%STRIPE-S-BOUNDLOG, stripe set STA8888: defined as STRIPE$TEST
%STRIPE-I-CLUSTER_SUPPORT, DEVICE  CLUSTER WIDE AFTER BINDING ON EACH NODE
%STRIPE-I-WSYNDEQ, Waiting for synch-lock to dequeue
  Testing STRIPE LIST command...
 Starting data integrity test: initializing stripe volume...
%MOUNT-I-MOUNTED, STRIPE$TEST  mounted on _STA8888:
    Configuration for stripe device _STA8888:
  Local cluster access available
  Number of units:                      2
    Units:
  $1$DUA210: IVP$_TOPS10_ (AIM1, AIM3) errcnt:0
  $1$DUA211: IVP$_TOPS20_ (READY, AIM3) errcnt:0
  Blocks/chunk:                         69
  Blocks/trip:                          138
  Overhead file system size:            2001
  Largest unsegmented IO size (blks)    138
Statistics for stripe device STRIPE$TEST
  Collection start time                  4-FEB-1991 14:36:29.59
  Dynamic memory in use:                60024
  Maximum queue length:                 1
  <= 1 chunk transfers:                 698
  <= 1 trip transfers:                  2822
  > 1 trip transfers:                   635
  Aligned transfers > 1 trip:           5
  Unaligned transfers > 1 trip:         630
```

Example 3–2 Cont'd on next page

Example 3–2 (Cont.) Sample Verification

```
  IOs split by member units:           0
  Read operations:                     4677
  Write operations:                    5234
  Blocks read:                         145444
  Blocks written:                      148057
  Average number of blocks read:       31
  Average number of blocks written:    28
  Average Bytes/sec 6261248  IOs/sec   412.95
  Average queue depth                  0.0
  Transfers blocked for no memory:     0
  Transfers blocked for no STRP:       0
 Device is present on the following nodes:
 AIM1,
  Writing test data to stripe device...
  Test data written to stripe device now starting compare.
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSEXE]ACC.EXE;1
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSEXE]ACLEDT.EXE;1
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSEXE]AGEN$FEEDBACK.EXE;1
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSEXE]ANALAUDIT.EXE;1
 .
 .
 .
 (Approximately four pages omitted from this sample listing.)
        .
 .
 .
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSLIB]XDPS$DPSBINDINGSSHR.EXE;1
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSLIB]XDPS$DPSCLIENTSHR.EXE;1
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSLIB]XDPS$DPSLIBSHR.EXE;1
%BACKUP-S-COMPARED, compared STRIPE$TEST:[SYSLIB]XNL$SHR.EXE;1
  STRIPE LIST after transfers
 Configuration for stripe device _STA8888:
  Local cluster access available
  Number of units:                     2
    Units:
  $1$DUA210: IVP$_TOPS10_ (AIM1, AIM3) errcnt:0
  $1$DUA211: IVP$_TOPS20_ (READY, AIM3) errcnt:0
  Blocks/chunk:                        69
  Blocks/trip:                         138
  Overhead file system size:           2001
  Largest unsegmented IO size (blks)   138
Statistics for stripe device STRIPE$TEST
  Collection start time                 4-FEB-1991 14:36:54.64
  Dynamic memory in use:               60024
  Maximum queue length:                1
  <= 1 chunk transfers:                5736
  <= 1 trip transfers:                 2816
  > 1 trip transfers:                  638
  Aligned transfers > 1 trip:          7
  Unaligned transfers > 1 trip:        631
  IOs split by member units:           0
  Read operations:                     4325
  Write operations:                    4865
  Blocks read:                         145162
  Blocks written:                      147765
  Average number of blocks read:       33
  Average number of blocks written:    30
  Average Bytes/sec 148480  IOs/sec    9.9
  Average queue depth                  0.0
  Transfers blocked for no memory:     0
  Transfers blocked for no STRP:       0
Device is present on the following nodes:
 AIM1,
%STRIPE-I-DEASSIGNED, deassigned system logical name STRIPE$TEST
```

Example 3–2 Cont'd on next page

Example 3–2 (Cont.) Sample Verification

```
%STRIPE-I-DISMOUNTED, dismounted volume on unit _AIM1$DUA210:
%STRIPE-I-DISMOUNTED, dismounted volume on unit _READY$DUA211:
%STRIPE-S-UNBOUND, stripeset STRIPE$TEST unbound

  The verification testing of STRIPE IFT V2.0 has completed successfully
```

4 Stripeset Commands

This chapter documents the four stripeset commands. The commands are sequenced alphabetically. You can remove these pages and file them in the *VAX/VMS DCL Dictionary* if you desire.

The four stripeset commands are:

- STRIPE BIND
- STRIPE INITIALIZE
- STRIPE LIST
- STRIPE UNBIND

STRIPE BIND

Binds a list of member disks into a stripeset virtual device[1].

FORMAT

STRIPE BIND *unit-list [logical-name]*

restrictions

This command requires PHY_IO and SYSNAM privileges.

PARAMETERS

unit-list

Specifies the device names of the disks that will be members of the stripeset. This parameter is a list of device names separated by a comma. The disk devices may or may not be mounted.

[logical-name]

Specifies a systemwide logical name to be assigned to the stripeset virtual device. *Digital recommends that you use the logical name option to associate a systemwide logical name with each stripeset virtual device.*

DESCRIPTION

The STRIPE BIND command binds a list of member disks into a stripeset and then binds the stripeset into a stripeset virtual device.

On a VAXcluster system, the STRIPE BIND command must be entered *on each node* needing local access to the stripeset.

The STRIPE BIND command mounts *each member unit* (MOUNT/SYSTEM) in order to access the overhead file system on the member. The STRIPE BIND command consults the overhead file system data written on the member units by STRIPE INITIALIZE, and confirms that the specified volumes form a consistent stripeset.

After a STRIPE BIND operation, the VMS software views the resultant stripeset virtual device as only another disk drive to which all valid VMS disk management commands apply.

Note that the STRIPE BIND command does **not** mount the *stripeset virtual device itself.* The DCL INITIALIZE and MOUNT commands may be applied to the stripeset virtual device once a BIND operation succeeds. However, neither the DCL INITIALIZE nor MOUNT commands are required; the stripeset virtual device supports applications that operate on unmounted and/or foreign mounted disks.

If a STRIPE BIND command is entered for a stripeset that is already bound and a logical name is specified, that logical name will be assigned to the existing stripeset. This enables an MSCP served stripeset virtual device to be bound to a logical name on any VMS Version 5.3 or earlier VAXcluster node, regardless of which VAXcluster node is serving the stripeset virtual device.

[1] Disks comprising the stripeset must have previously been given the STRIPE INITIALIZE command. Before issuing the STRIPE BIND command, you must issue the STRIPE INITIALIZE command to each disk in the stripeset.

QUALIFIERS

/ALLOCATION_CLASS=

Sets the allocation class for all stripesets bound on this node. The range for /ALLOCATION_CLASS is from 1 to 255.

The first BIND command sets the allocation class for all stripesets bound on that node whether or not /ALLOCATION_CLASS is used.

If /ALLOCATION_CLASS is not used, the stripeset virtual device is given a generic device name with no allocation class (for example, STA1).

A stripeset virtual device must be known by the same allocation class to all VAXcluster nodes which access it.

Plan and set your allocation classes carefully, because changing an allocation class usually requires a system reboot.

/DEVICE_NAME=

Assigns a device name to the stripeset virtual device (in the format of STA*xx*, *xx* being a number from 0 to 9999).

A stripeset virtual device must be known by the same device name to all nodes which access it. For instance, do not use *STA12* on one node and *5STA12* on another.

If you fail to use the /DEVICE_NAME qualifier, the virtual device is assigned a default device name of STA9998, assigned in descending numerical order as each new stripeset is created. If the STRIPE UNBIND command is used to take a stripeset virtual device off line, the assigned default name STA9998 will again be available for use.

/SYSTEM (D)
/NOSYSTEM

Specifies whether or not stripeset member units are mounted with global access for users on the system. The default is /SYSTEM.

If /NOSYSTEM is specified, when the process that entered this command is terminated, the stripeset members will be dismounted and the stripeset will go into mount verification.

If you are using /NOSYSTEM with shadowset members that are no longer in a shadowset, you must mount the shadowset members with the following command:

```
$ MOUNT/NOSYSTEM/OVER=SHADOW_MEMBERSHIP $1$DUAxx
$ MOUNT/NOSYSTEM/OVER=SHADOW_MEMBERSHIP $1$DUAyy
```

Then enter the STRIPE BIND command:

```
$ STRIPE BIND/NOSYSTEM $1$DUAxx, $1$DUAyy
$_  /DEVICE_NAME=STAzz
```

STAzz must not be on line prior to entering the STRIPE BIND command on the VAXcluster nodes.

/TRANSFER_BLOCK=

Specifies an upper limit for the largest unsegmented I/O that can be transferred through the virtual device. Default size is 127 blocks. Performance may be improved for applications that use I/O requests greater than 127 blocks. See Section 2.6.1 for further information.

EXAMPLES

1 `$ STRIPE BIND DUA1:,DUA2: DISK$STRIPE`

This command binds two devices (DUA1: and DUA2:) into a stripeset and assigns the stripeset the logical name of DISK$STRIPE.

2 `$ STRIPE BIND DUA1:,DUA2:,DUA3:/DEVICE_NAME=STA1`

This command binds three devices (DUA1:, DUA2:, and DUA3:) into the stripeset STA1 and does not assign it a logical name.

STRIPE INITIALIZE

Defines the membership and permanent characteristics of a stripeset.

FORMAT

STRIPE INITIALIZE *unit_list volume_id_list*

restrictions

This command requires PHY_IO and SYSNAM privileges.

The units specified in the *unitlist* must not mounted.

PARAMETERS

unit_list

Specifies the device names of the disks that will be members of the stripeset. This parameter is a list of device names separated by a comma. The disk devices must not be mounted.

volume_id_list

Specifies the volume labels that are assigned to each of the units in the *unitlist*. The number of volume labels must equal the number of units specified in the *unitlist*. The order of the comma-separated volume labels must correspond to that of the units.

DESCRIPTION

The STRIPE INITIALIZE command defines the membership and permanent characteristics of a stripeset. It records this information on the member volumes in a fashion similar to the DCL INITIALIZE command.

Any data formerly on a member volume is lost when the STRIPE INITIALIZE command is entered.

By default, the STRIPE INITIALIZE command will ask for confirmation before initializing the member disks.

STRIPE INITIALIZE mounts each member unit and writes the stripeset overhead file system on each one. STRIPE INITIALIZE leaves the member units mounted; however, the virtual device itself is not mounted.

QUALIFIERS

/CHUNK_SIZE=n

Determines the mapping between the stripeset virtual device and its members.

LBNs are assigned to member units in chunks of size *n*. That is, the first *n* data blocks are assigned to the first unit on the *unit_list*, the next *n* data blocks are assigned to the next unit on the *unit_list*, and so on.

If /CHUNK_SIZE is not specified, STRIPE INITIALIZE uses the number of blocks per track of the member units as a default. Test results show in the majority of cases, this default is optimal. **Give Careful consideration to using anything other than the default setting.**

For information on setting chunk_size, consult the *VAX Disk Striping Driver Guide to Performance and Tuning*.

/CONFIRM (D)
/NOCONFIRM

If /CONFIRM is specified, STRIPE INITIALIZE prompts for confirmation that the correct units and volumes were selected. STRIPE INITIALIZE requires a YES response from SYS$COMMAND prior to writing on the member volumes. Any other response aborts the operation.

If /NOCONFIRM is specified, INITIALIZE will **not** prompt for confirmation, regardless of what is on the member volumes.

/CONFIRM is the default.

/FILES_11 (D)
/NOFILES_11

/FILES_11 qualifier limits the size of the stripeset to 8.5 GB (16,777,215 blocks). /FILES_11 is the default.

See Section 2.6, Managing Stripesets Larger Than 8.5 GB, for more information.

EXAMPLES

1
```
$  STRIPE INITIALIZE/CONFIRM DUA1:,DUA2:,DUA3: INV1,INV2,INV3


%STRIPE-I-VOLID, unit _DUA1: contains volume STRP1
%STRIPE-I-VOLID, unit _DUA2: contains volume STRP2
%STRIPE-I-VOLID, unit _DUA3: contains volume STRP3
Proceed to initialize specified units? [Y/N] (N) :   Y
```

This command defines a stripeset that consists of units DUA1:, DUA2:, and DUA3: with corresponding volume labels INV1, INV2, and INV3. The system will prompt for confirmation.

Since no /CHUNK_SIZE was specified, the stripesets' chunk size will be set to the number of blocks per track of the member units.

STRIPE LIST

Reports the configuration and performance data of a stripeset virtual device.

FORMAT

STRIPE LIST *Device_name*

restrictions

There are no restrictions.

PARAMETERS

Device_name

The device name of the stripeset virtual disk or an equivalent logical name.

DESCRIPTION

The STRIPE LIST command lists the characteristics of the specified stripeset virtual device. This includes a list of all the member units, overhead file system data, and performance statistics.

A typical terminal output of the STRIPE LIST command is shown in the following example.

QUALIFIERS

/OUTPUT=file-spec

Specifies that the output of the command is to be directed to the specified file. If this qualifier is not specified, the output is directed to SYS$OUTPUT.

/ZERO

/ZERO qualifier updates the start time of data collection and resets all stripeset counters. Use of this qualifier enables you to reset counters for performance monitoring/testing.

EXAMPLES

1

```
$  STRIPE LIST/OUTPUT=STA20.DAT $5$STA20
```

This command places the configuration and performance data of stripeset 5STA20 into the file STA20.DAT.

2 `$ STRIPE LIST $5$STA20`

This command directs the configuration and performance data of stripeset 5STA20 to the SYS$OUTPUT device. An example of the output is:

```
Configuration for stripe device _$5$STA20:
  Local cluster access available
  Number of units:                      2
    Units:
  DUA64: STRP64 (HSC013, PICKUP) errcnt:0
  DUA65: STRP65 (HSC013, PICKUP) errcnt:0
  Blocks/chunk:                         31
  Blocks/trip:                          62
  Overhead file system size:            1829
  Largest unsegmented IO size (blks)    1000
Statistics for stripe device STA20
  Collection start time                 15-OCT-1990 12:54:49.82
  Dynamic memory in use:                59972
  Maximum queue length:                 1
  <= 1 chunk transfers:                 140
  <= 1 trip transfers:                  0
  > 1 trip transfers:                   0
  Aligned transfers > 1 trip:           0
  Unaligned transfers > 1 trip:         0
  IOs split by member units:            0
  Read operations:                      60
  Write operations:                     80
  Blocks read:                          60
  Blocks written:                       80
  Average number of blocks read:        1
  Average number of blocks written:     1
  Average Bytes/sec 0  IOs/sec          0.1
  Average queue depth                   0.0
  Transfers blocked for no memory:      0
  Transfers blocked for no STRP:        0
Device is present on the following nodes:
```

STRIPE UNBIND

Sets the stripeset virtual device off line and frees the associated member units.

FORMAT

STRIPE UNBIND *Device_name*

restrictions

This command requires PHY_IO and SYSNAM privileges.

The stripeset virtual device must be dismounted before being unbound. If the stripeset virtual device is mounted at the time the STRIPE UNBIND command is entered, the command will fail.

PARAMETERS

Device_name

The device name of the stripeset virtual disk or an equivalent logical name.

DESCRIPTION

The STRIPE UNBIND command sets the specified stripeset virtual device off line and frees the member units. The member units are left dismounted. The data stored on the member disks is not affected.

If you unbind a stripeset and do not disturb the data on the member units, you can rebind the stripeset with the data intact as it was before the unbind operation.

EXAMPLES

1
```
$  DISMOUNT STA20
```

This command dismounts stripeset STA20 in preparation for the STRIPE UNBIND command.

2
```
$  STRIPE UNBIND STA20
%STRIPE-I-DISMOUNTED, dismounted volume on unit _DUA1:
%STRIPE-I-DISMOUNTED, dismounted volume on unit _DUA2:
%STRIPE-S-UNBOUND, stripeset STA20 unbound
```

This command deletes stripeset STA20 and frees the member units DUA1: and DUA2:. The member units are left dismounted.

5 STRIPE LIST Command Usage

The STRIPE LIST command can perform as a data gathering tool within a specified timeframe, thus allowing a snapshot view of the software's operation. The IO$_STRIPE_LIST can be used as a callable interface to the STDRIVER, allowing you to write a customized program for performance monitoring. This chapter addresses both uses.

5.1 The STRIPE LIST Command Output

There are several ways to monitor a stripeset. You may use the VMS MONITOR utility, VAX Performance Advisor software, VAX SPM (Software Performance Monitor) software, and/or the STRIPE LIST command. Here, we will describe the fields of the STRIPE LIST command.

5.1.1 STRIPE LIST Fields

- **Local cluster access available–**Describes the environment the stripeset is running in; a VAXcluster environment running VMS Version 5.4 or greater that enables nodes local cluster access to this Version 2.0 stripeset. If software versions are earlier than VMS Version 5.4 or Disk Striping Version 2.0, this field will appear as ''Served cluster access available,'' and will enable VAXcluster nodes served cluster access to the stripeset.
- **Number of units–**Total number of members that make up the stripeset.
- **Units–**Lists the member's name, volume id, error count, and SCS node names of the controllers with access to the member.
- **Blocks/chunk–**Displays the number of blocks [1] that make up a chunk for this stripeset. Chunk size is the number of blocks written to one member before mapping to the next member.
- **Blocks/trip–**Displays the trip size in units of logical blocks. A trip is the number of blocks needed to write one chunk of data to all members of the stripeset once.
- **Overhead file system size–**The number of logical blocks required, per stripeset member, to store the overhead file system.
- **Largest unsegmented I/O size in blocks–**The largest I/O request that the VMS software will pass to the stripeset without breaking the I/O into smaller pieces.

[1] One block=512 bytes.

- **Dynamic memory in use–**Amount of nonpaged pool (in bytes) currently in use. This is not reset by the STRIPE LIST/ZERO command.
- **Collection start time–**This time reflects when data collections began. This field, along with the following fields' counters, is updated by the STRIPE BIND command or by the STRIPE LIST/ZERO command.

Note: Data collection begins at the date and time displayed in the ''Collection start time'' field. The following fields' counters are returned to zero by the STRIPE LIST/ZERO command.

- **Maximum queue length–**Maximum number of simultaneously outstanding I/O requests to the stripeset.
- **<= 1 chunk transfers–**The number of I/O requests to the stripeset that were satisfied by I/O to a single member.
- **<= trip transfers–**The number of I/O requests whose size was less than or equal to one trip.
- **> 1 trip transfers–**The number of I/O requests to the stripeset that were decomposed into multiple member requests, but were less than, or equal to, one trip.
- **Aligned transfers > 1 trip–**The number of page-aligned I/O requests whose size was greater than one trip.
- **Unaligned transfers > 1 trip–**The number of nonpage-aligned I/O requests whose size was greater than one trip.
- **I/Os split by member units–**This field is incremented every time an I/O request to a stripeset member is only partially completed. In this case, the stripeset must complete the request by issuing additional I/Os to the member until the transfer is complete. It is, therefore, possible that a single I/O request to a stripeset member may increment this counter multiple times. This field has the possibility of incrementing when the stripeset member is using an SSP controller.
- **Read operations–**The number of stripeset I/O requests that were read operations.
- **Write operations–**The number of stripeset I/O requests that were write operations.
- **Blocks read–**The number of blocks that have been read from the stripeset.
- **Blocks written–**The number of blocks that have been written to the stripeset.
- **Average number of blocks read–**The total number of blocks read divided by the total number of read operations.
- **Average number of blocks written–**The total number of blocks written divided by the total number of write operations.
- **Average bytes/sec I/Os/sec–** Total bytes transferred by the stripeset divided by the length of the collection interval; total I/Os completed by the stripeset divided by the length of the collection interval.

- **Average queue depth–**The average number of I/O requests outstanding to the stripeset, when a new I/O request is received.
- **Transfers blocked for no memory–**The number of I/O requests that were stalled due to a shortage of nonpaged pool.
- **Transfers blocked for no STRP–**The number of I/O requests that were stalled due to the shortage of internal stripeset data structures. Please submit a Software Performance Report (SPR) (along with a copy of the STRIPE LIST command) if this counter gets incremented.
- **Device is present on the following nodes–**Lists the SCS node names of the VAXcluster nodes on which this stripeset has been bound. In a single CPU system with no node name, this list will be empty. If one or more of the nodes has left the VAXcluster, without first unbinding the stripeset, this list may be inaccurate.

5.2 QIO Interface for IO$_STRIPE_LIST

The IO$_STRIPE_LIST function code allows users to access the dynamic statistics that are collected by the stripeset. This can also be displayed by using the STRIPE LIST command. For a more complete description of these items, refer to Section 5.1.1.

IO$_STRIPE_LIST and its item codes are defined in the system macro $STRIPEDEF. This macro is inserted into the library STARLET.MLB during installation.

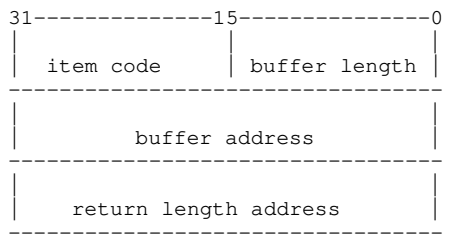
The format of the QIO is:

$QIO [W] [efn] , chan, func, [,iosb] [,astadr] [,astprm],P1

where:

Qualifier	Meaning
efn	event flag number
chan	a short word containing the assigned channel number
func	IO$_STRIPE_LIST
iosb	two long words used for I/O status block
astadr	*ast* address to return to upon completion if so desired
astprm	a value passed to the *ast* routine
P1	ITMLIS, a VMS ITEM_LIST_3 format

The form of the ITEM_LIST_3 is:



Item Code—A word containing a user-supplied symbolic code specifying the item of information that IO$_STRIPE_LIST is to return. The $STRIPEDEF macro defines these codes.

Buffer Address—A longword containing the user-supplied address of the buffer in which IO$_STRIPE_LIST is to write the information.

Return Length Address—A longword containing the user-supplied address of a word in which IO$_STRIPE_LIST writes the length (in bytes) of the information actually returned.

Table 5–1 IO$_STRIPE_LIST Item Codes

Item Codes	Definitions
STRIPE$_UNITS	Number of members in the stripeset. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_CHUNK_SIZE	Chunk size (in LBNs) specified for the stripeset. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_RELOCATION	Size of overhead file system in LBNs. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_MEMORY_USE	Amount of nonpaged pool currently in use (in bytes). This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_MEMSHIP_PTR	List of nodes that have the stripeset bound. This returns an ASCII string of the SCS node names separated by a comma and a space. The buffer should be a minimum of 2176 bytes long, so the buffer length field should contain a minimum of 2176 bytes.
STRIPE$_LTIME	64-bit time that collections started. This is standard VMS 64-bit time that is returned. The buffer length should be 8 bytes to accommodate the time.
STRIPE$_ZERO	Sets collection counters to zero and resets collection start time. This function does not return any values.

Table 5–1 (Cont.) IO$_STRIPE_LIST Item Codes

Item Codes	Definitions
STRIPE$_MAX_STRP_USE	Maximum number of Stripeset I/O Request Packets (STRPs) in use since the incarnation of the stripeset. This is returned as a longword, so the buffer length field should be four bytes.
STRIPE$_MUL_THREAD	Contains the cumulative number of outstanding I/O requests since time collections began. To calculate the average queue depth, this value should be divided by the total number of I/O requests. This item returns a longword, so the buffer length field should be four bytes.
STRIPE$_STRP_USE	Current number of STRPs in use; a striping IRP. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_ONE_MEMBER_XFRS	Number of transfers that involved only one member of the stripeset. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_LE_TRIP_XFRS	Number of transfers that involved more than one member of the stripeset up to the total number of members, but never visiting one member more than once. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_GT_TRIP_XFRS	Number of transfers that involve at least one member of the stripeset more than once. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_ALIGNED_XFRS	Number of transfers greater than a trip whose buffer is aligned on a page (512 bytes) boundary. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_UNALIGNED_XFRS	Number of transfers greater than a trip whose buffer is not aligned on a page (512 bytes) boundary. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_SPLIT_IOS	Number of I/O requests sent to member units that were split into smaller I/Os by the member units. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_READ_OP_COUNT	Number of read operations. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_WRITE_OP_COUNT	Number of write operations. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_BLOCKS_READ	Number of blocks read. This is returned as a longword, so the buffer length field should contain four bytes.

Table 5–1 (Cont.) IO$_STRIPE_LIST Item Codes

Item Codes	Definitions
STRIPE$_BLOCKS_WRITTEN	Number of blocks written. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_NO_MEM_XFRS	Number of times sufficient dynamic memory was unavailable for STDRIVER. In this case, the STDRIVER is blocked for memory resources. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_NO_STRP_XFRS	Number of times STDRIVER needed more STRPs than were preallocated. In this case, the STDRIVER is blocked waiting for an STRP. This is returned as a longword, so the buffer length field should contain four bytes.
STRIPE$_LAST_STRP_XFRS	Number of times STDRIVER used all the STRPs that were preallocated. This is returned as a longword, so the buffer length field should contain four bytes.

Note that additional information can be calculated from the values shown in Table 5–1. For example, the average I/O size can be determined by multiplying the number of blocks by 512 bytes and dividing the result by the number of I/O operations.

6 Striping Internals

This chapter provides more detail about how the VAX Disk Striping Driver software functions.

6.1 Stripeset Logical Block Mapping

The VAX Disk Striping Driver software, or STDRIVER, can be used to create one or more virtual disk devices called **stripesets**. A stripeset consists of two or more physical disks called **stripeset members**, or simply *members.*

Each VMS compatible disk device can be considered an array of logical blocks. Physical disk device drivers and controllers map these logical blocks onto physical sectors that are organized in concentric tracks and cylinders, but these are largely hidden from the software which uses the disk device.

A stripeset can also be considered an array of logical blocks. But, the VAX Disk Striping Driver software maps each logical block of a stripeset onto a logical block of one of its members. This mapping is fixed when the stripeset is initialized, and is controlled by a single parameter called the *chunk size*. Understanding the mapping between a stripeset and its members is critical to understanding the operation of a stripeset.

6.1.1 Stripeset Member Structure

Initializing a stripeset divides the logical blocks of each member into three adjacent partitions:

- the **overhead file system partition**,
- the **data partition**, and
- the **replacement control table partition**, or **RCT partition**.

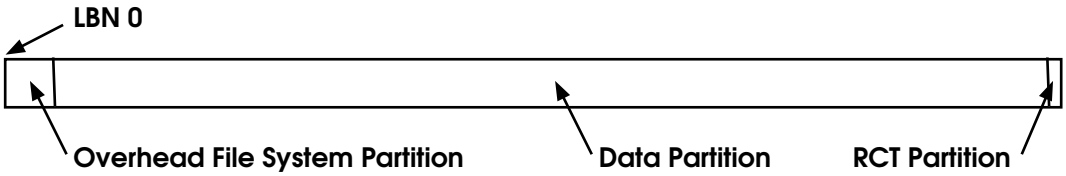
Stripeset initialization creates a Files-11 file system on each member in which all reserved files and root directory files are contained within the overhead file system partition. Two additional files are created in the root directory. Data that describes the structure and state of the stripeset are stored in STRIPE$VOLS.SYS, and the data and RCT partitions are mapped by STRIPE$DATA.SYS. The size of the overhead file system partition must be large enough to store a minimal Files-11 file system, but its size depends on the disk geometry. For instance, an RA90 disk drive overhead file system partition consumes 2001 blocks (29 tracks).

Although the overhead file system can be mounted, its use is reserved for STRIPE commands. Customers **should not use utilities to modify files or change the placement of files in the overhead file system**. Specifically, defragmenting files in the overhead file system may result in corrupted data.

A portion at the end of each stripeset member is reserved for the RCT partition, as shown in Figure 6–1, Stripeset Member Logical Block Space. The RCT partition of the first member is used to store a replacement control table, or RCT. Each DSA disk includes an RCT, and some VMS utilities access this structure. Because a stripeset of DSA disks appears as a DSA disk, the stripeset must contain a trivial RCT for those VMS utilities which expect one.

The logical blocks contained in the data partitions of the stripeset members are mapped into stripeset logical blocks. The data partition begins on a track boundary.

Figure 6–1 Stripeset Member Logical Block Space



6.1.2 Stripeset Virtual Disk Structure

There is a one-to-one correspondence between each logical block of a stripeset and each logical block in the data partitions of the stripeset members. Thus, the size of a stripeset virtual disk is the sum of the sizes of the member data partitions.[1]

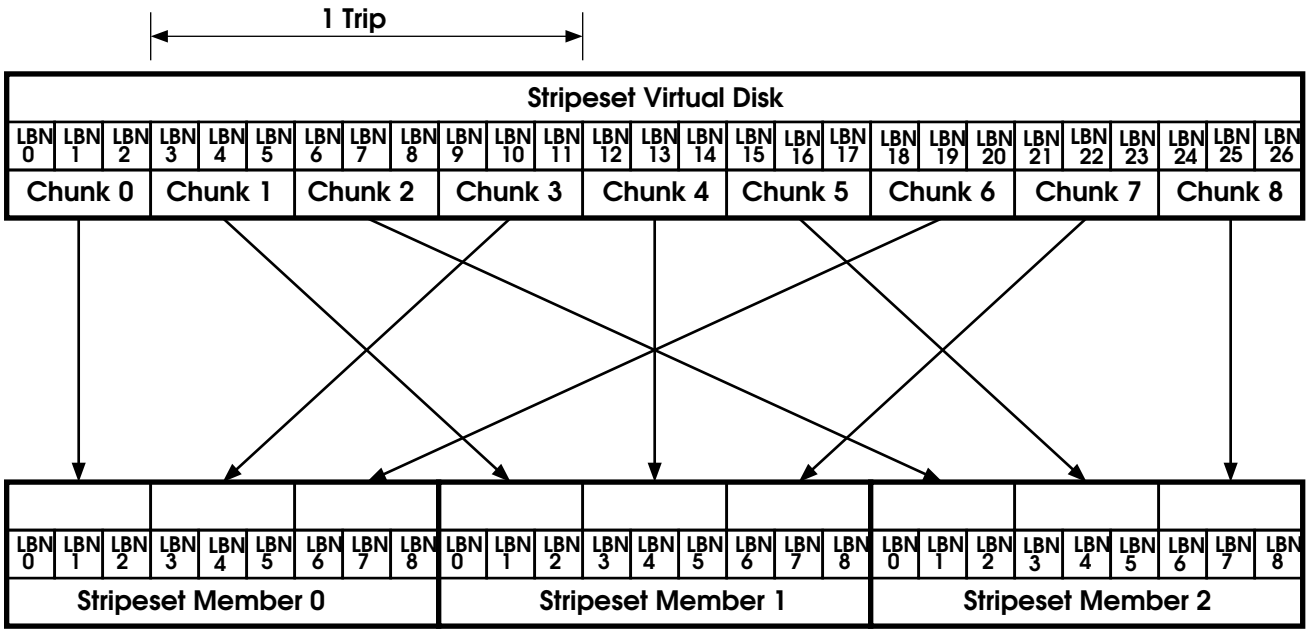
The logical blocks of a stripeset are grouped into equal size **chunks**. The logical blocks within a chunk map to consecutive logical blocks on a single stripeset member, and consecutive chunks map to the members of the stripeset in a round-robin fashion.

If a stripeset consists of *n* members, then the stripeset's **trip size** is its chunk size times *n*; and a trip is any *n* consecutive chunks. Because the member data partition size is chosen so that each member stores an equivalent integer number of chunks, the stripeset virtual disk consists of an integer number of trips.

The following figure, Figure 6–2, Mapping Stripeset Logical Blocks to Member Logical Blocks describes the relationship between logical blocks, chunks, and trips for a 3-member stripeset with chunk size equal to three.

[1] Because all members of a stripeset must be identical, the capacity of a stripeset is the number of members times the size of the general member data partition less the size of the RCT partition.

Figure 6–2 Mapping Stripeset Logical Blocks to Member Logical Blocks



Note: Stripeset member logical block numbers (LBNs) are shown relative to the start of the data partition.

6.2 Stripeset Data Flow

Although applications may use high-level language read and write commands, the result of every record and file I/O request is ultimately one or more disk $QIO system service calls, or simply **QIOs**. Each disk QIO specifies that a group of consecutive logical blocks be written, read, or erased to or from a disk. The $QIO system service has arguments which, among other things, specify:

- size of the transfer,
- logical block address of the start of the transfer, and
- virtual address of the memory buffer.

Between the time that a request is issued and the time that it is complete, a QIO is considered to be *outstanding*. The number of QIOs that are outstanding to a disk at a given time is called the disk **queue length**.

When an I/O request is issued to a stripeset, the $QIO system service dispatches the request to the STDRIVER. The STDRIVER then processes the request in a way which depends on the size of the transfer.

6.2.1 Small Transfers

When the requested data resides in a single chunk, then the STDRIVER simply dispatches the request to the physical disk driver which corresponds to the stripeset member which maps that chunk. Of course, the logical block address of the start of the transfer must be modified according to the stripeset mapping.

For example, consider the stripeset depicted in Figure 6–2, Mapping Stripeset Logical Blocks to Member Logical Blocks, and suppose that a request is made for two blocks starting at LBN 7. Because LBNs 7 and 8 correspond to LBNs 1 and 2 of stripeset member 2, this request is converted to a 2-block request starting at LBN 1 and then is dispatched to member 2.

While an I/O request is outstanding to the stripeset member, it also appears to be outstanding to the stripeset virtual disk. However, during this time other requests may be processed by the STDRIVER because it is multithreaded.

6.2.2 Medium Size Transfers

When the requested data resides in two or more chunks, then the STDRIVER decomposes the request into multiple requests and dispatches those to the physical disk drivers for the stripeset members which map those chunks. Such a request remains outstanding to the stripeset virtual disk until all of the corresponding requests to the member disks complete. However, this does not prevent other stripeset requests from being processed in the meantime.

In this case, the STDRIVER must modify the starting logical block address of each of the member requests, but it must also modify the transfer length and memory buffer addresses of the member requests.

Again, consider the stripeset in Figure 6–2, Mapping Stripeset Logical Blocks to Member Logical Blocks. But, this time suppose that a request is made for six blocks starting at LBN 8. When decomposed, this corresponds to three member requests:

- a 1-block transfer starting at LBN 1 of member 2 (stripeset LBN 8),
- a 3-block transfer starting at LBN 3 of member 0 (stripeset LBNs 9, 10, and 11), and
- a 2-block transfer starting at LBN 3 of member 1 (stripeset LBNs 12 and 13).

In this case, the 3-member disk may concurrently satisfy the dispatched requests.

6.2.3 Large Transfers

When more than one trip of data is requested, the simple decomposition described in the previous section might cause a member disk to receive two I/O requests resulting from the same stripeset I/O request. To improve efficiency, the STDRIVER minimizes the number of stripeset member I/O requests by combining requests where possible. As a result, a single stripeset I/O request will never result in more than one I/O request to each of its members.

Again, consider the stripeset in Figure 6–2, Mapping Stripeset Logical Blocks to Member Logical Blocks. Suppose a request is made for twelve blocks starting at LBN 3. When decomposed, this would correspond to four member requests:

- a 3-block transfer starting at LBN 0 of member 1 (stripeset LBNs 3-5),
- a 3-block transfer starting at LBN 0 of member 2 (stripeset LBNs 6-8),
- a 3-block transfer starting at LBN 3 of member 0 (stripeset LBNs 9-11), and
- a 3-block transfer starting at LBN 3 of member 1 (stripeset LBNs 12-14).

Note that the first and last of these member requests are for consecutive blocks on member 1. So, the STDRIVER combines these into a single request for six blocks starting at LBN 0 of member 1.

Because the virtual addresses of the memory buffers of the two parts of the transfer that is directed to member 1 are not contiguous, the STDRIVER must take special measures in order to combine those transfers.

When the user's *memory buffer is aligned on a page boundary*, the STDRIVER uses page table entries to map the discontiguous portions of the user's memory buffer into a virtually contiguous buffer in the system virtual address space. That is, portions of the user's memory buffer are double mapped while the stripeset I/O request is outstanding. This method provides very efficient data transfer but may require a large number of page table entries.

When the user's *memory buffer is not aligned on a page boundary* and the transfer size is larger than one trip, the STDRIVER allocates a buffer in nonpaged pool large enough to accommodate the transferred data. MOVC3 is used to move the data between the user's buffer and this nonpaged pool buffer. Because the nonpaged pool buffer is page-aligned, the above mapping scheme can be used to avoid additional I/O operations as the data is moved between the nonpaged pool buffer and the stripeset members. Substantial additional space in nonpaged pool, system page table entries and CPU cycles can be required for such a transfer.

A STRIPE Command Error Messages

This appendix contains the messages that are issued by and unique to the STRIPE command. Refer to the *VMS System Messages and Recovery Procedures Reference Manual* for explanations of messages issued by the VMS system. Refer to the *Guide to VMS Software Installation* for explanations of messages issued by VMSINSTAL.

A.1 Message Format

The messages documented in this appendix will appear on your screen in the following format:

```
%STRIPE-s-mnemonic, text
-STRIPE-s-mnemonic, text
```

where:

%	is the prefix to all primary messages.
–	is the prefix to any continuation messages.
STRIPE	is the source designation of the message.
s	is the severity level of the message.
mnemonic	is an abbreviation of the message text.
text	is the expanded text of the message.

Severity Level

The severity level of the STDRIVER messages is one of the following:

Code	Meaning
S	Success—Successful completion of the request.
I	Informational—May or may not require user action.
W	Warning—Request may not have completed and may require user action.
E	Error—System encountered an error which may be recoverable.
F	Fatal—System encountered a fatal error and cannot continue processing this request.

A.2 STRIPE Messages

The following message descriptions are alphabetized by the mnemonic portion of the message. The message prefix, source designation, and severity code are not shown.

ASNFAIL, cannot assign *ddcu*, STDRIVER probably not loaded,

Explanation: The STRIPE BIND command was unable to assign an I/O channel to the stripeset virtual disk.

User Action: Have your system manager execute the command file SYS$STARTUP:STRIPE$STARTUP.

BADCHUNK, invalid /CHUNK_SIZE value *nnn*,

Explanation: An invalid chunk size, such as zero or a negative value, was specified on the STRIPE INITIALIZE command line.

User Action: Reenter the command and specify the chunk size as a positive integer that is less than the total size of the stripeset.

BADNAME, _1STA0: Name does not match existing _STA0:Name,

Explanation: Attempt to bind an existing stripeset using a /DEVICE_ NAME or /ALLOCATION_CLASS different from that which was originally specified.

User action: If you are binding an existing stripeset, but are on a different node, you must use the same device name. If you are creating a different stripeset, then the device name must be unique.
In either case, issue the STRIPE BIND command with /DEVICE_ NAME=STRIPESET_NAME. For example:

```
/DEVICE_NAME=sta0.
```

BINDFAIL, bind operation failed,

Explanation: The STRIPE BIND operation failed due to a condition described in a continuation message.

User Action: Refer to the continuation message for user action. For example:

```
-SYSTEM-F-NOPRIV, no privilege for attempted operation.
 Attempt to bind stripeset without phys_io and sysnam privileges.
 (User action) See your system manager for privileges.
```

BOUND, stripe-set *physical_name* defined,

Explanation: This is an informational message confirming that the physical STA device was created.

User Action: None.

BOUNDLOG, stripe-set *physical_name* defined as *logical_name*,

Explanation: This is an informational message confirming that the physical STA device was created and the stripeset was assigned the logical name supplied in the STRIPE BIND command line.

User Action: None.

BTRANSBLKSZ, invalid /unsegmented io size value *nnn*,

Explanation: Attempt to bind a stripeset with a transfer_block size that is not in the range of 127 to 2 billion blocks.

User action: Reenter STRIPE BIND command with /TRANSFER_ BLOCK in the range of 127 to 2 million blocks. Increasing this parameter may require an increase in the SYSGEN parameter SPTREQ. See Section 3.1.6, Adjusting SYSGEN Parameters.

CLUSTER, unsuitable cluster factor,

Explanation: The requested cluster size is too small for the stripeset virtual disk, or no /CLUSTER_FACTOR was specified, and the default (default = 3) is too small.

User Action: Calculate the minimum cluster factor *n*, using the formula below. Then, with the result, reenter the VMS INITIALIZE/CLUSTER_ FACTOR=n command.

$$\frac{number\ of\ members \times total\ LBNs\ per\ member}{255 \times 4096} = clusterfactor(n)$$

DEASSIGNED, deassigned system logical name *logical_name*,

Explanation: This is an informational message confirming that the logical name previously assigned by the STRIPE BIND command was deassigned by the STRIPE UNBIND command.

User Action: None.

DEVOFFLINE, Device is not in configuration or not available,

Explanation: Attempt to list a stripeset that is off line (unbound).

User action: None.

DISMOUNTED, dismounted volume on unit *ddcu*,

Explanation: This is an informational message confirming that the specified volume was dismounted during the STRIPE UNBIND operation.

User Action: None.

DISMOUNTFAIL, failed to DISMOUNT member unit *ddcu*,

Explanation: STRIPE UNBIND failed to dismount the specified volume because of a condition described in a continuation message.

User Action: Refer to the continuation message for user action.

DRVNOLOAD, STDRIVER not loaded,

Explanation: Attempt to bind a stripeset when STDRIVER has not been loaded by SYSGEN.

User action: To correct, have your system manager execute the command file SYS$STARTUP:STRIPE$STARTUP

GETDVIWFAIL, GETDVIW failed,

Explanation: STDRIVER was unable to obtain device or volume information on the specified STA device during a STRIPE LIST, STRIPE BIND, or STRIPE UNBIND operation.

User Action: Submit a Software Performance Report (SPR).

HOMEFAIL, error reading home block,

Explanation: An error occurred while reading the home block of a member device during a STRIPE BIND operation. This home block was presumably read correctly by the STRIPE INITIALIZE command.

User Action: Retry the STRIPE BIND command. Getting the same error indicates an unrecoverable failure of the member device.

INIT, initializing member unit *ddcu* as volume *volid*,

Explanation: This is an informational message confirming that STRIPE INITIALIZE is initializing this member device with the given volume label.

User Action: None.

INITFAIL, INITIALIZE failed for unit *ddcu*,

Explanation: Some part of the device initialization (such as device allocation) failed during the STRIPE INITIALIZE operation.

User Action: Check the state of the specified unit to determine the cause (for example, is the unit already allocated to another user; already bound). See accompanying message for information on *which* unit failed to initialize.

INSFSPTS, insufficient SPTs to map process buffer to system,

Explanation: There are insufficient system page table entries to map a process buffer to the system.

User Action: Section 3.1.6, Adjusting SYSGEN Parameters explains the number of system page table entries required by the STRIPE BIND command. Either increase the SYSGEN parameter SPTREQ, or decrease the /TRANSFER_BLOCK qualifier.

INVALCOM, invalid command,

Explanation: Occurs if an invalid STRIPE command is entered.

User Action: Use one of the four STRIPE commands: STRIPE INITIALIZE, STRIPE BIND, STRIPE UNBIND, or STRIPE LIST.

LISTFAIL, STRIPE_LIST failed,

Explanation: The STRIPE LIST command failed, unable to successfully complete a QIO with the IO$_STRIPE_LIST function code. The stripeset may be unbound, and off line.

User Action: If the stripeset virtual device was on line, submit an SPR.

LOGFAIL, logical name *logical_name* not defined,

Explanation: The STRIPE BIND command failed, unable to create the specified logical name.

User Action: Refer to the continuation message for user action.

NAMUSED, the name ''STAxx'' is already used by a different stripeset,

Explanation: A STRIPE BIND command has been entered with the same /DEVICE_NAME as an existing stripeset, although with different members.

User Action: Reenter the command using a /DEVICE_NAME not currently in use.

MOUNTFAIL, failed to MOUNT member unit *ddcu*,

Explanation: The VMS MOUNT command failed because of a condition specified in a continuation message.

User Action: Refer to the continuation message for user action.

MOUNTING, mounting volume *volid* on unit *ddcu*,

Explanation: This is an informational message confirming that the STRIPE INITIALIZE or STRIPE BIND command is mounting the specified unit.

User Action: None.

NOCONFIRM, command not confirmed, operation aborted,

Explanation: An inappropriate response was entered to the STRIPE INITIALIZE/CONFIRM prompt.

User Action: Use Y[es], N[o], or Return when you respond to this prompt.

NOCREPRC, failed to create subprocess needed to implement command,

Explanation: A call to LIB$SPAWN failed.

User Action: Ensure that the subprocess quota is equal to or greater than two.

NOPRIV, No privilege for attempted operation,

Explanation: Attempt to enter one of the STRIPE commands without the required privileges.
1. STRIPE INITIALIZE, STRIPE BIND and STRIPE UNBIND require PHY_IO and SYSNAM privileges.
2. STRIPE LIST requires the PHY_IO privilege unless the STRIPE.EXE image has been installed with PHY_IO privilege.

User action:
1. See your system manager for privileges.
2. Have your system manager execute @SYS$STARTUP:STRIPE$STARTUP.COM

NOTAVAIL, unit *ddcu* not available for striping,

Explanation: The specified unit cannot be accessed or is inappropriate for use by the STDRIVER software, such as a device that is already mounted or allocated to another user. A continuation message provides additional information.

User Action: Refer to the continuation message for user action.

NOTSTA, device *ddcu or _DUA0:* is not a stripeset device,

Explanation: A STRIPE command was attempted on a device other than a stripeset virtual device (for example, STRIPE LIST DUA1).

User Action: Specify the stripeset name on the command line.

NOUNITS, no stripeset units specified,

Explanation: No units were specifed on the STRIPE INITIALIZE or STRIPE BIND command line.

User Action: Reenter the command line and specify the desired units.

NOVOL, unit *ddcu* volume has no FILES-11 volume label,

Explanation: An error occurred while reading the homeblock on the specified member unit during a STRIPE BIND command. This will occur on an attempt to STRIPE BIND a stripeset that has not been prepared with STRIPE INITIALIZE. It will also occur if the stripeset volume is corrupted because of failed hardware or software.

User Action: If the list of member disks is incorrect, reenter STRIPE BIND. Otherwise, enter the STRIPE INITIALIZE command and reenter the STRIPE BIND command. Note that INITIALIZE will destroy all data presently on the disks.

SIZELIMIT, STAxx has been limited to a size of 8.5 GB.,

Explanation: Indicates that the stripeset size is limited to 8.5 GB. See Section 2.6.

User Action: None.

SSINCONST, STRIPE$VOLS.SYS format error or inconsistency,

Explanation: The STRIPE BIND command detected one or more missing, different, or extra member units that were not defined as members of a stripeset in the previous STRIPE INITIALIZE command.

User Action: Reenter the STRIPE BIND command specifying the member units defined in the corresponding STRIPE INITIALIZE command.

STAMNT, _STA0: MUST BE DISMOUNTED BEFORE UNBINDING.,

Explanation: Attempt to unbind a stripeset that is still mounted.

User action: Dismount stripeset; then unbind.

STANOBND, _STA0: STRIPE SET IS NOT BOUND,

Explanation: Attempt to unbind a stripeset that is not bound.

User action: No action; stripeset is already unbound.

TOOSMALL, a unit is too small to use for striping,

Explanation: Attempt to STRIPE INITIALIZE a stripeset consisting of members that are too small to be useful. This occurs if the minimal overhead file system occupies more than 5% of each member disk.

User Action: This will only occur on diskettes. Avoid striping diskettes.

ULINCON, unit list inconsistency,

Explanation: Two identical unit names were entered on the STRIPE BIND command.

User Action: Reenter the STRIPE BIND command and specify unique unit names.

UNBINDFAIL, UNBIND failed for Stripeset,
-SYSTEM-F-NOPRIV, no privilege for attempted operation.

Explanation: Attempt to stripe unbind without phys_io and sysnam privileges.

User action: See your system manager for privileges.

UNBOUND, stripe-set *physical_name* unbound,

Explanation: This is an informational message confirming that the physical STA*n* device was unbound.

User Action: None.

UNSUPFEATURE, FUNCTION NOT ALLOWED WITH THIS VERSION,

Explanation: Attempt to bind an existing stripeset with a VAX Disk Striping Driver V1.0 format on more than one node in a VAXcluster system.

User action: Upgrade your Version 1.0 stripeset to a Version 2.0 stripeset as described in Section 3.1.7.

VNF, unable to access ddcu:[000000]STRIPE$VOLS.SYS,

Explanation: Attempt to bind a volume that has not been STRIPE INITIALIZED.

User action: Check to see if volumes used in STRIPE BIND command have been STRIPE INITIALIZED and that the STRIPE BIND command has been entered correctly.

VOLID, unit *ddcu* contains volume *volid*,

Explanation: This is an informational message by the STRIPE INITIALIZE operation reporting that the specified member unit contains a Files-11 volume with the specified Volume ID.

User Action: Ensure that you intend to initialize this volume as part of the stripeset. When initialized, any data on this volume will be lost.

VOLTOOSHORT, not enough volumes specified—must match number of units,

Explanation: The STRIPE INITIALIZE operation detected fewer Volume IDs than device names on the STRIPE INITIALIZE command line.

User Action: Reenter the STRIPE INITIALIZE command and specify an equal number of Volume IDs and device names.

WBIND, waiting for synch-lock to bind device,

Explanation: Informational message noting another STRIPE BIND or UNBIND command is currently executing on the cluster. Your current BIND procedure will wait until the previous STRIPE commands finish.

User Action: None.

WSYNDEQ, waiting for synch-lock to dequeue,

Explanation: After the BIND or UNBIND command, several locks must be dequeued. There may be a slight pause while this process occurs.

User Action: None.

WUNBIND, waiting for synch-lock to bind device,

Explanation: Informational message noting another STRIPE BIND or UNBIND command is currently executing on the cluster. Your current BIND procedure will wait until the previous STRIPE commands finish.

User Action: None.

XTRAVOL, too many volumes specified—must match number of units,

Explanation: The STRIPE INITIALIZE operation detected more Volume IDs than device names on the STRIPE INITIALIZE command line.

User Action: Reenter the STRIPE INITIALIZE command and specify an equal number of Volume IDs and device names.

Glossary

chunk: A set of consecutive stripeset logical blocks that map to a single stripeset member disk.

chunk size: The number of logical blocks in a chunk (see chunk).

data transfer rate: The speed at which data is transferred through a device or a bus, usually measured in megabytes per second (MB/sec).

disk striping: A parallel decomposition technique for disk drives that allows multiple disk drives to concurrently process a single I/O request.

Files-11: The name of the disk file structure used by the VMS operating system.

gigabyte, or GB: One billion bytes, or 10^9 bytes. A measured unit of storage comprised of bits.

home block: A data block in a disk volume's index file that contains information about the volume as a whole, such as volume label and protection.

I/O request rate: The number of individual I/O requests that can be processed in a unit time, usually measured in I/O requests per second.

ISE: Integrated Storage Elements. DSSI disk and tape devices which consist of an integrated drive and controller and are known as ISEs.

LBN: Logical Block Number. A volume-relative address of a data block on a mass storage device.

License Management Facility: The License Management Facility, or LMF. Software in the VMS operating system that helps a system administrator conform to licensing restrictions encoded in Product Authorization Keys (PAKs).

LMF: See License Management Facility.

load balancing: Dividing the demand among disks so that no one disk is over-utilized.

MSCP: Mass Storage Control Protocol. A message level protocol by which a host computer expresses I/O commands for a DSA subsystem.

MSCP served disk: A locally connected disk (such as a stripeset, XMI, BI, MASSBUS, or UNIBUS disk) that is made available to other VAXcluster members by means of the Mass Storage Control Protocol server running on its CPU.

member unit: One of the disk devices that make up a stripeset.

member volume: One of the disk volumes that make up a stripeset.

nonpaged pool: Dynamically allocated memory, managed by the VMS operating system, which can never be paged or swapped. This is a finite resource established at SYSBOOT time and used by the disk striping driver.

overhead file system: The area on each member unit containing the stripeset database. This includes the stripeset's identification and members' volume labels.

PAK: See Product Authorization Key.

page-aligned buffer: An I/O buffer whose address is zeros in its nine low-order bits. The use of page-aligned I/O buffers minimizes disk striping overhead when the data transfer size is greater than one trip.

Product Authorization Key: A license, issued to a customer, that contains codes that authorize the use of a software product. Disk striping will not operate without a valid PAK registered with LMF. PAKs may have expiration dates or other restrictions. (See *VMS License Management Utility Manual* and *A Practical Guide to Managing Software Licenses*.)

QIO: Queue Input/Output. The VMS system service that prepares an I/O request for processing by the device driver. A QIO is synonymous with an I/O request in VMS software.

RMS: Record Management Services. A set of operating system procedures, called by programs to process files and records within files.

SCSI: Small Computer Systems Interface.

SPT: System Page Table. A data structure that maps the system region virtual addresses.

SSP: Storage system port. SSP controllers are operated with a PUDRIVER and include UDA50, KDA50, KDB50, KDM70, KFQSA, and RQDX3 controllers.

stripeset: A set of disk drives operating in concert as a single virtual disk (pseudodevice) to provide increased I/O performance.

A set of disk volumes initialized with the information required to function in a stripeset.

structure level: The version number of Files-11 used to initialize a particular volume.

trip: The span of the largest single striped transfer which accesses each member exactly once.

trip size: The number of LBNs in a trip, equal to the chunk size times the number of members.

unit: A disk drive.

VPA: The VAX Performance Advisor, a Digital software product that assists a system manager in monitoring the performance of a VMS system, identifies system problems and their causes, and recommends corrective actions.

volume label: The volume identifier (up to 12 alphanumeric characters) in the home block of a Files-11 disk file.

Index

A

Allocation class • 2–8
  how to set • 2–8
AUTOGEN
  using • 3–4

B

Binding stripeset members • 2–2

C

Calculating
  minimum cluster size • 2–3
CLUSTER_SIZE qualifier • 2–3
Configuration
  examples • 2–5
  planning • 3–1
Configuring
  stripesets • 2–1
  stripesets on a VAXcluster • 2–1
Creating a stripeset
  on a single CPU • 2–1
  on a VAXcluster • 2–10
Creating a VMS file system on a stripeset • 2–3

D

Defining a stripeset virtual device • 2–1
Disk striping
  benefits • 1–4
    data transfer rates • 1–4
    load balancing • 1–5
    throughput • 1–6
      taking advantage of • 1–6
  reliability • 1–6
  shadow sets • 2–15

E

Error messages • A–1
Examples
  configuration
    MSCP serving a stripeset • 2–5
    striping MSCP served disks • 2–5
  creating a stripeset on a VAXcluster system • 2–11
  installation verification • 3–9
  mapping LBNs to stripeset member units • 1–3
  mapping LBNs to Stripeset member units • 1–3
  single CPU stripeset creation • 2–4
  STDRIVER installation • 3–6

F

Figures
  configuration
    MSCP serving a stripeset • 2–5
    striping MSCP served disks • 2–5
FILES-11 file system
  creating on a stripeset • 2–3

I

Initializing stripeset members • 2–1
Installation
  procedure • 3–6
Installation verification • 3–9
Installing STDRIVER • 3–1

L

License PAK
  registering • 3–5
License Product Authorization Key • 3–5
  see license PAK • 3–5
Load balancing • 1–5
  taking advantage of • 1–4, 1–5

M

Maintaining a stripeset • 2–19
  backing up shadow sets • 2–19
Mapping
  logical block numbers • 1–2
  logical blocks • 1–3
MONITOR POOL
  using • 3–4
Mounting a VMS file system on a stripeset • 2–3
MSCP server
  enabling the • 2–9
  serving the stripeset members • 2–5
  serving the stripeset virtual device • 2–5

P

Post-installation activities • 3–7
Preinstallation
  procedures • 3–2
    registering license PAK • 3–5
    software upgrade • 3–5
  requirements • 3–2
  restrictions • 3–2
Preinstallation planning • 3–1
  configuration • 3–1
Procedures
  creating a stripeset
    on a single CPU • 2–1
    on a VAXcluster • 2–10
  installation • 3–6
  post-installation • 3–7
  preinstallation • 3–2
    registering license PAK • 3–5
  software upgrade • 3–5
  volume shadowing and striping on a single CPU • 2–13
  volume shadowing in a striped VAXcluster environment • 2–14

R

Registering the license PAK • 3–5

S

Sample installation verification • 3–9
Sample STDRIVER installation • 3–6
Shadowed disk
  failover using VAXsimPLUS • 2–16
SPT entries
  allocation • 3–3
Statistics
  items reported • 5–4
STDRIVER
  adjusting SYSGEN parameters • 3–3
    memory requirements • 3–4
    SPT entries • 3–3
  files added to system • 3–9
  I/O decomposition • 1–2
  installation • 3–1
  installation procedure • 3–6
  installation verification • 3–9
  mapping LBNs • 1–2
  post-installation
    procedure • 3–7
  post-installation activities • 3–7
  preinstallation • 3–1
    requirements • 3–2
    restrictions • 3–2
  preinstallation planning • 3–1
  sample installation • 3–6
  sample verification • 3–9
  software upgrade procedure • 3–7
    preparation • 3–5
  startup file modifications • 3–8
STRIPE BIND command • 2–2, **4–3**
STRIPE command
  messages • A–1
STRIPE INITIALIZE command • 2–1, **4–6**
STRIPE LIST command • **4–8**
  output • 5–1
  output field definitions • 5–3
Stripeset
  allocation class • 2–8
  chunk size • 1–3
  configuration
    MSCP serving • 2–5
    MSCP serving strategy • 2–5
    planning for • 2–5
  creating a VMS file system • 2–3
  creation on a VAXcluster • 2–10
  creation on a VAXcluster prior to VMS Version 5.4 • 2–11

Stripeset (Cont.)
data collection • 5–1
data flow • 6–3
multiple chunk transfers • 6–4
multiple trip transfers • 6–5
single chunk transfers • 6–4
defining the virtual device • 2–1
maintenance • 2–19
member structure • 6–1
data partition • 6–1
example • 6–2
overhead file system • 6–1
replacement control table • 6–1
memory buffers
nonpage aligned • 6–5
page aligned • 6–5
mounting a VMS file system • 2–3
reliability
increasing • 1–6
restrictions
quorum disk • 2–17
size • 2–17
system disk • 2–17
system dump file • 2–17
size • 2–17
structure
chunks • 6–2
example • 6–2
logical blocks • 6–2
trips • 6–2
virtual disk • 1–2
maximum transfer size • 2–17, 2–18
volume shadowing • 2–13
a served VAXcluster node • 2–15
controller-based • 2–13
host-based • 2–13
in a VAXcluster environment • 2–14
Stripeset commands • **4–1**
STRIPE BIND • 2–2, **4–3**
STRIPE INITIALIZE • 2–1, **4–6**
STRIPE LIST • **4–8**
STRIPE UNBIND • 2–16, **4–10**
Stripeset member units
binding • 2–2
failure notification using VAXsimPLUS • 2–16
initializing • 2–1
unbinding • 2–16
STRIPE UNBIND command • 2–16, **4–10**
Striping shadow sets • 2–13
SYSGEN parameters, adjusting
AUTOGEN
using • 3–4
memory requirements • 3–4
MONITOR POOL
using • 3–4
SPT entries • 3–3
SYSGEN problems
detecting • 3–4
SYSTEM resources
calculating • 3–4

T

Throughput
taking advantage of • 1–6

U

Unbinding stripeset members • 2–16
Upgrading
STDRIVER software, Version 1.0 • 3–5
Upgrading STDRIVER software • 3–7

V

VAXsimPLUS • 2–16
shadowed disk failover • 2–16
Volume shadowing
and striping in a single CPU environment • 2–13
a served VAXcluster node • 2–15
in a striped VAXcluster environment • 2–14