DIGITAL StorageWorks
HSZ50 Array Controller

HSOF 5.1
Configuration Manual

Part Number: EK–HSZ50–CG. C01

March 1997

Software Version: **HSOF Version 5.1**

Digital Equipment Corporation
Maynard, Massachusetts

March 1997

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense. Restrictions apply to the use of the local-connection port on this series of controllers; failure to observe these restrictions may result in harmful interference. Always disconnect this port as soon as possible after completing the setup operation. Any changes or modifications made to this equipment may void the user's authority to operate the equipment.

Warning!
This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Achtung!
Dieses ist ein Gerät der Funkstörgrenzwertklasse A. In Wohnbereichen können bei Betrieb dieses Gerätes Rundfunkstörungen auftreten, in welchen Fällen der Benutzer für entsprechende Gegenmaßnahmen verantwortlich ist.

Avertissement!
Cet appareil est un appareil de Classe A. Dans un environnement résidentiel cet appareil peut provoquer des brouillages radioélectriques. Dans ce cas, il peut être demandé à l' utilisateur de prendre les mesures appropriées.

Table of Contents

Index

Glossary

Figures

Tables

Related documents

The following table lists some of the documents related to the use of this product.

Document title	Part number
DECevent Installation Guide	AA–Q73JA–TE
StorageWorks BA350–MA Controller Shelf User's Guide	EK–350MA–UG
StorageWorks Configuration Manager for DEC OSF/1 Installation Guide	AA–QC38A–TE
StorageWorks Configuration Manager for DEC OSF/1 System Manager's Guide for HSZterm	AA–QC39A–TE
StorageWorks Solutions Configuration Guide	EK–BA350–CG
StorageWorks Solutions Shelf and SBB User's Guide	EK–BA350–UG
StorageWorks Solutions SW300-Series RAID Enclosure Installation and User's Guide	EK–SW300–UG
StorageWorks SW500-Series Cabinet Installation and User's Guide	EK–SW500–UG
StorageWorks SW800-Series Data Center Cabinet Installation and User's Guide	EK–SW800–UG
The RAIDBOOK—A Source for RAID Technology	RAID Advisory Board
Polycenter Console Manager User's Guide	Computer Associates
VAXcluster Systems Guidelines for VAXcluster System Configurations	EK–VAXCS–CG
16-Bit SBB User’s Guide	EK-SBB16-UG
7-Bit SBB Shelf (BA356 Series) User’s Guide	EK-BA356-UG
SBB User’s Guide	EK-SBB35-UG

1

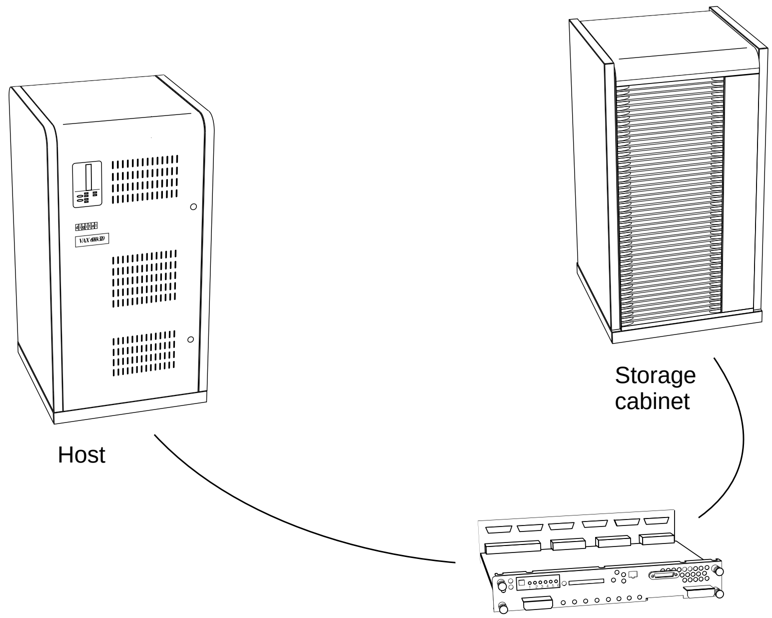
Introduction

Features of your controller

Your HSZ50 controller is the intelligent bridge between your host and the devices in your subsystem.

From the host’s perspective, the controller is simply another SCSI –2 device connected to one of its I/O buses. Consequently, the host sends its I/O requests to the controller just as it would to any SCSI –2 device.

Figure 1—1 Bridging the gap between the host and its storage subsystem



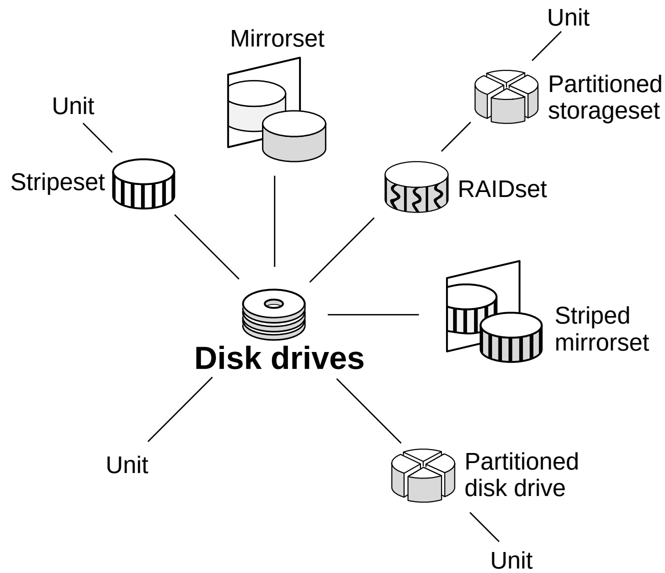
From the subsystem’s perspective, the controller receives the I/O requests from the host and directs them to the devices in the subsystem. Since the controller processes all of the I/O requests, it eliminates the host-based processing that’s typically associated with reading and writing data to multiple storage devices.

But the controller does much more than simply manage I/O requests: it provides the ability to combine several ordinary disk drives into a single, high-performance storage unit called a storageset.

Storagesets are implementations of RAID technology, also known as a “Redundant Array of Independent Disks.” Every storageset shares one important feature: whether it uses two disk drives or ten, each storageset looks like a single storage unit to the host.

You create storage units by combining disk drives into storagesets, such as stripesets, RAIDsets, and mirrorsets, or by presenting them to the host as single-disk units as shown in Figure 2.

Figure 1—2 The host recognizes units created from storagesets, partitions, and disk drives



- **Stripesets** (RAID 0) combine disk drives in serial to increase transfer or request rates.
- **Mirrorsets** (RAID 1) combine disk drives in parallel to provide a highly reliable storage unit.
- **RAIDsets** (RAID 3/5) combine disk drives in serial—just like stripesets—but also store parity data to ensure high reliability.
- **Striped mirrorsets** (RAID 0+1) combine mirrorsets in serial to provide the highest throughput and availability of any storage unit.

Of course, the controller also lets you add tape drives, loaders, and libraries to your subsystem to meet all of your storage requirements.

Table 1—1 Summary of features

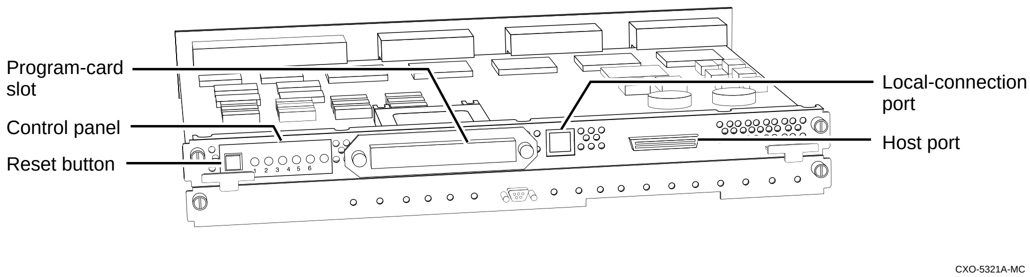
This table summarizes the features of your controller:

Feature	Supported
Host bus interconnect	SCSI–2 FWD
Host protocol	SCSI–2
Device protocol	SCSI–2
Number of SCSI device ports	6
Number of SCSI-2 devices in single configuration	42
Number of SCSI-2 devices in dual-redundant configuration	36
RAID levels	0, 1, 0+1, 3/5
Cache size	32, 64, or 128 MB
Preferred target IDs	up to 4
PCMCIA updates	✓
Device warm swaps	✓
Exercisers for testing devices	✓
Tape drives, loaders, and libraries	✓
Number of configuration entities (devices + storagesets + partitions + units)	195
Maximum number of storagesets	30
Maximum number of partitions per storageset or disk drive	4
Maximum number of RAIDsets and mirrorsets running simultaneously	20
Maximum number of units presented to host	32
Maximum number of devices per unit	32
Largest device, storageset, or unit	120 GB

Anatomy of your controller

Take a few moments to familiarize yourself with the controller’s components shown in Figure 1—3.

Figure 1—3 Key components



Under normal circumstances, you won’t need to remove the controller from its cabinet. For this reason, the components that you’ll use most often are conveniently located on the front panel. For example, the local-connection port provides a convenient way to connect a terminal to your controller so that you can configure it for the first time.

After you’ve configured your controller, you should periodically check its control panel. The reset button flashes green about once every second to indicate that the controller is operating normally. If an error occurs, one or more of the amber LEDs on the control panel will flash in a pattern that will help you to diagnose the problem.

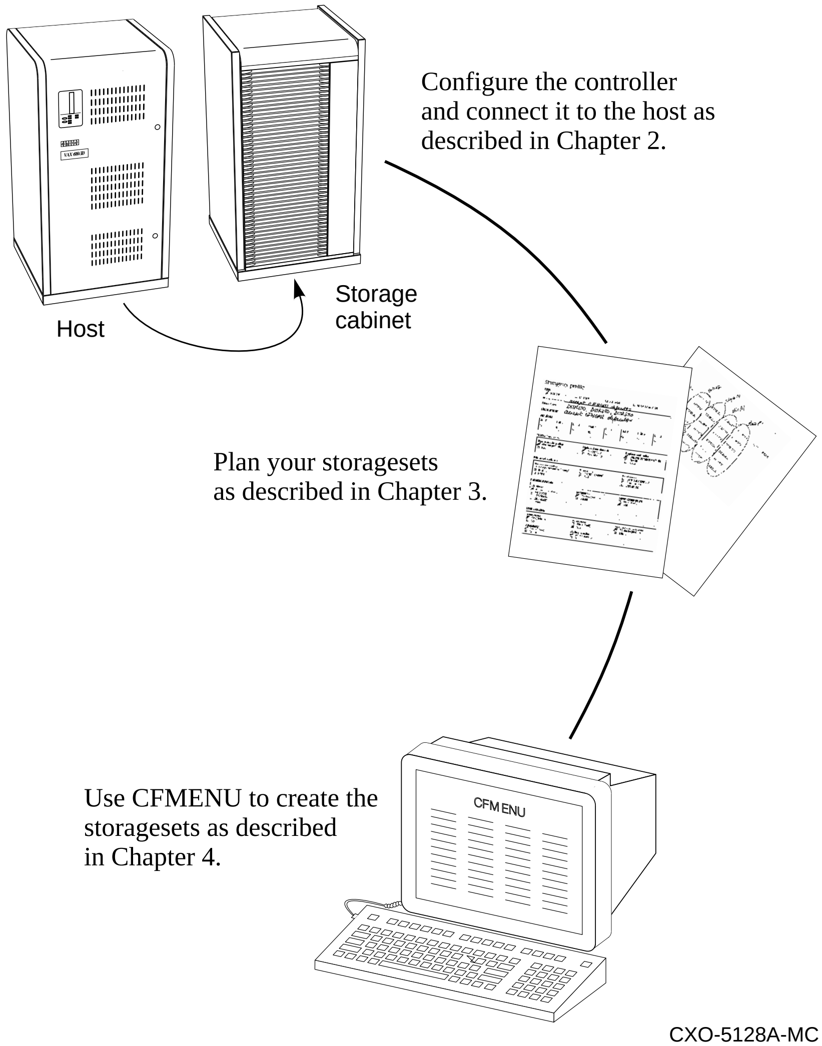
The host port and program-card slot are also located on the front panel, making it easy to update the HSOF software or to connect the controller to a different host.

The backplane enables two controllers to communicate with each other in dual-redundant configurations. It also contains device ports that enable the controller to communicate with the devices in your subsystem.

Key steps for configuring your subsystem

Figure 1—4 shows the key steps you'll follow to set up and configure your subsystem and its controller. Each of these key steps is explained later in this book.

Figure 1—4 Key steps for configuring your subsystem



2

Configuring your controller

Key steps for configuring a controller

Communicating with a controller

Configuring a controller

Connecting a controller to the host

Key steps for configuring a controller

Unless you specifically ordered a preconfigured subsystem, you will have to configure your controller and its subsystem before you can use them. Follow these key steps to configure a controller:

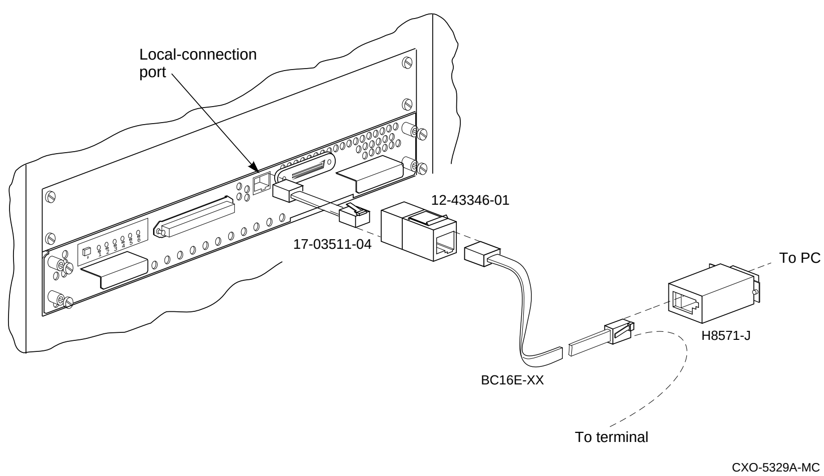
1. Establish communications with the controller. If you are configuring a controller for the first time, you MUST use a local connection.
2. Configure the controller.
3. Connect the controller to the host.

After you have configured your controller, you can configure your subsystem by creating storagesets, single disk units, and other storage devices as described in Chapters 3, 4,
and 5.

Communicating with a controller

You can communicate with a controller locally or remotely. Use a local connection to configure the controller for the first time. Use one of the remote connections described in the appendicies for all subsequent configuration tasks.

Figure 2—1 Connect your terminal to the local-connection port to set the controller's initial configuration



Caution

The local-connection port described in this book generates, uses, and can radiate radio-frequency energy through cables that are connected to it. This energy may interfere with radio and television reception. Do not leave any cables connected to it when you are not communicating with the controller.

To establish a local connection for setting the controller's initial configuration:

1. Turn off the terminal and connect it to the controller as shown in Figure 5. Plug one end of a DECconnect Office Cable (BC16E-XX) into the terminal; plug the other end into the adapter (12-43346-01); use the extension (17-03511-04) to connect the adapter to the controller's local-connection port.

   If you are using a PC instead of a terminal, you will need the serial-port adapter (H8571-J), also shown in Figure 5.
2. Turn on the terminal.
3. Configure the terminal for 9600 baud, 8 data bits, 1 stop bit, and no parity.
4. Press the Enter or Return key. A copyright notice and CLI prompt appear, indicating that you have established a local connection with the controller.

Configuring a controller

You can configure a controller to operate as a single controller or as one controller in a pair of dual-redundant or multiple bus failover dual-redundant controllers:

- Use a **single** (nonredundant) controller only if you want to use one controller to service the same group of storagesets, single -disk units, and other storage devices. Mount the controller in its own shelf and follow the steps for configuring a single controller that begin on page 2–4.
- If you have two controllers of the same type, running the same HSOF Version, you can configure them as **dual-redundant** controllers. Use this configuration if you want to use two controllers to service the same

group of storagesets, single-disk units, and other storage devices. Since both controllers service the same storage units, either controller can continue to service all of the units if the other controller fails.

Mount both controllers in the same shelf and follow the steps for configuring dual-redundant controllers that begin on page 2–5.

Note

Your host must have two SCSI adapters as well as operating-system software to support the multiple bus failover enhancement.

Multiple bus failover is a dual-redundant configuration in which each controller has its own separate connection to the host. Thus, if one of the controllers loses contact with the host, the other controller can service all of the storage units through its host connection until you repair the failed connection. Of course, since both controllers service the same storage units, either controller can continue to service all of the units if the other controller fails.

Mount both controllers in the same shelf and follow the steps for configuring multiple bus failover dual-redundant controllers that begin on page 2–7.

Configuring a single controller

To configure a single (nonredundant) controller:

1. Establish a local connection to the controller.
2. Set the SCSI target IDs for the controller:

   ```
   CLI> SET THIS_CONTROLLER ID = (n,n,n,n)
   ```

   where *n,n,n,n* represents from one to four SCSI target IDs (0–7) that are not already being used by other devices or hosts on the host SCSI bus.

   Using more than one target ID allows the controller to present more units to the host. Enclose multiple IDs in parentheses and separate each by a comma.

3. Optional: change the CLI prompt.

   ```
   CLI> SET THIS_CONTROLLER PROMPT = "new prompt"
   ```

where *new prompt* is a 1- to 16-character string that will appear as the prompt. For example, you could use the prompt to indicate the controller's name,
such as “HSZ> ”.

4. Optional: set the maximum data-transfer rate.

```
CLI> SET THIS_CONTROLLER
TRANSFER_RATE_REQUESTED=speed
```

where *speed* is 10MHZ (default), 5MHZ, or ASYNCHRONOUS.
Table 2—1lists the maximum transfer rates for different lengths of fast and slow SCSI buses. These lengths represent cable lengths plus shelf-bus lengths.

Table 2—1 Maximum data transfer rates for different SCSI-bus cable lengths

Bus type	Transfer rate	Meters	Feet
8-bit, single ended	5 MHz	6	19.68
8-bit, single ended	10 MHz	3	9.84
16-bit, differential	20 MHz	25	82.02

Note

5 MHz is often referred to as SCSI 1.

5. Restart the controller:

```
CLI> RESTART THIS_CONTROLLER
```

6. When the CLI prompt reappears, verify the configuration:

```
CLI> SHOW THIS_CONTROLLER FULL
```

7. Connect the controller to the host.

Configuring dual-redundant controllers

To configure a pair of dual-redundant controllers:

Note

This procedure assumes that one controller has already been setup and installed, as described in the previous section.

---

1. Establish a local connection to one of the controllers. (For the steps that follow, the controller to which you are connected is "this controller.")
2. Put "this controller" into dual-redundant (failover) mode:

   ```
   CLI> SET FAILOVER COPY = THIS_CONTROLLER
   ```

   The "other controller" inherits "this controller's" configuration, then restarts. Wait for it to return to normal operation before continuing.
3. Declare up to four SCSI target IDs for the dual-redundant pair. Use the same SCSI target IDs for each controller:

   ```
   CLI> SET THIS_CONTROLLER ID = (n,n,n,n)
   CLI> SET OTHER_CONTROLLER ID = (n,n,n,n)
   ```

   where *n,n,n,n* represents the SCSI target IDs (0–7) that are not already being used on the host SCSI bus.

   Using more than one target ID allows the controllers to present more units to the host. Enclose multiple IDs in parentheses and separate each by a comma.
4. Prefer some or all of the SCSI target IDs to "this controller." The "other controller" automatically inherits the remaining IDs. During normal operation, each controller services only those storage units that are associated with its preferred IDs:

   ```
   CLI> SET THIS_CONTROLLER PREFERRED_ID = (n,n)
   ```

   where *n,n* is a subset of the SCSI target IDs that you declared in the previous step. Enclose multiple IDs in parentheses and separate them by a comma.

   Use preferred IDs to balance the I/O load among the storage units and thereby improve the throughput for the dual-redundant pair.

   You can also use the PREFERRED_ID switch to effectively make the "other controller" a hot standby by declaring that it has no preferred SCSI target IDs:

   ```
   CLI> SET OTHER_CONTROLLER NOPREFERRED_ID
   ```

By declaring that it has no preferred IDs, the "other controller" will not respond to any SCSI target IDs on the host SCSI bus. Instead, "this controller" will process all I/O during normal operation.

5. Optional: change the CLI prompt for each controller:

```
CLI> SET THIS_CONTROLLER PROMPT = "new prompt"
CLI> SET OTHER_CONTROLLER PROMPT = "new prompt"
```

where *new prompt* is a 1- to 16-character string that will appear as the prompt. For example, you could use the prompt to indicate each controller's name, such as "HSZ_1> " and "HSZ_2> ".

6. Optional: set the maximum data-transfer rate for each controller. Use the same rate for both controllers:

```
CLI> SET THIS_CONTROLLER
TRANSFER_RATE_REQUESTED=speed
CLI> SET OTHER_CONTROLLER
TRANSFER_RATE_REQUESTED=speed
```

where *speed* is 10MHZ (default), 5MHZ, or ASYNCHRONOUS. Table 2 lists the maximum transfer rates for different lengths of fast and slow SCSI buses.

7. Restart the controllers:

```
CLI> RESTART OTHER_CONTROLLER
CLI> RESTART THIS_CONTROLLER
```

8. When the CLI prompt reappears, verify the configuration for each controller:

```
CLI> SHOW THIS_CONTROLLER FULL
CLI> SHOW OTHER_CONTROLLER FULL
```

9. Connect the controller to the host.

Configuring multiple bus failover, dual-redundant controllers

To configure a pair of multiple bus failover, dual-redundant controllers:

1. Establish a local connection to one of the controllers. (For the steps that follow, the controller to which you are connected is "this controller.")
2. Put "this controller" into multiple bus failover mode:

   ```
   CLI> SET MULTIBUS_FAILOVER COPY = THIS_CONTROLLER
   ```

   The "other controller" inherits "this controller's" configuration, then restarts. Wait for it to return to normal operation before continuing.
3. Follow the procedures from step three onward for configuring dual-redundant controllers.

Connecting a controller to the host

The controller's configuration determines how you will connect it to a host. Whether you are installing a new controller or moving an old one to a new location, you should always configure it, or at least set its SCSI target IDs, before connecting it to a host. Failure to do so may adversely affect the host or cluster.

Note

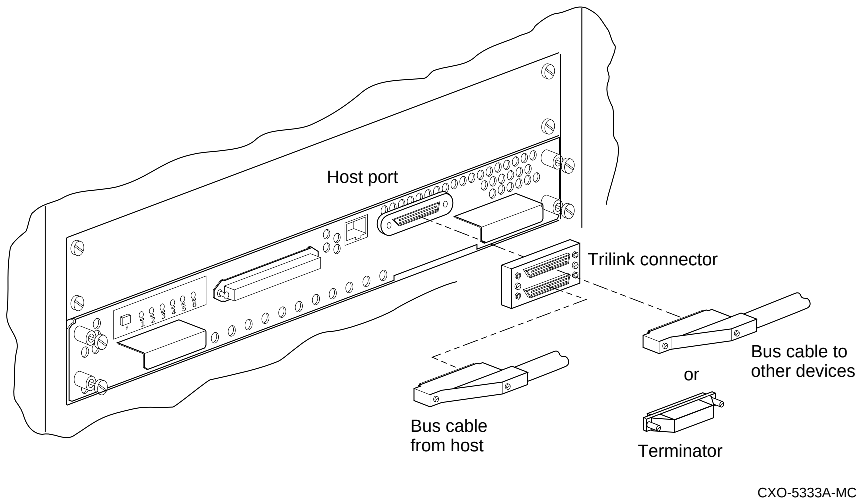
The controller's host port uses 16-bit FWD SCSI. If your host's SCSI adapter uses a different protocol, install a DWZZ-series SCSI-bus signal converter somewhere between the host and the bus cable that connects to the front of the trilink connector.

Follow one of these procedures to connect your controller to a host. Each of these procedures is described in detail in this section:

- Connecting a single (nonredundant) controller to the host.
- Connecting dual-redundant controllers to the host.
- Connecting multiple bus failover, dual-redundant controllers to the host.

Connecting a single controller to the host

Figure 2—2 Connecting a single controller to its host

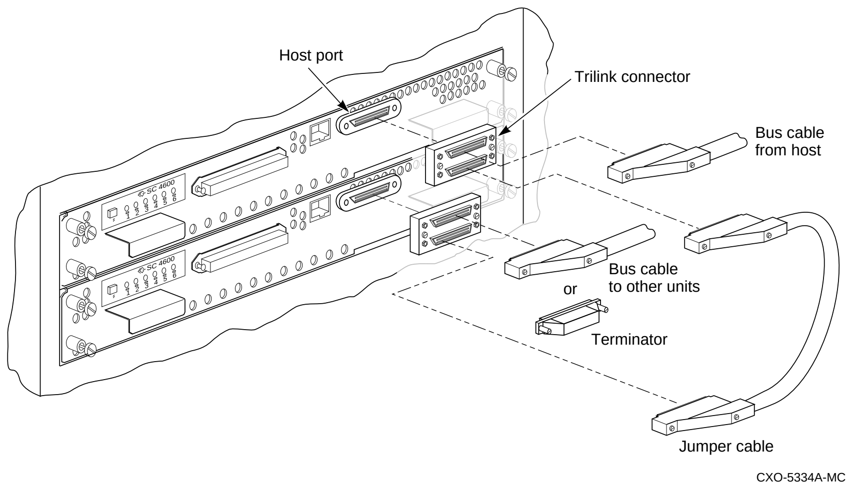


To connect a single, nonredundant controller to the host:

1. Stop all I/O from the host to its devices on the bus to which you are connecting the controller.
2. Remove the trilink connector (12-39921-01) from the controller. This connector is a 68-pin Y-adapter that maintains bus continuity even when it is disconnected from the controller.
3. Connect the bus cable from the host to one of the connectors on the front of the trilink connector as shown in Figure 2—1 Connect your terminal to the local-connection port to set the controller's initial configuration.
4. If the controller is at the end of the host bus, connect a terminator to the other connector on the front of the trilink connector. Otherwise, connect a cable that continues to the next device on the bus. (Be sure to install a terminator at the end of the bus.)
5. Reconnect the trilink connector to the host port on the controller. Do not disconnect the host cables from the trilink connector.
6. Route and tie the cables as desired.
7. Restart the I/O from the host. Some operating systems may require you to reboot the host to see the devices attached to the new controller.

Connecting dual-redundant controllers to the host

Figure 2—3 Connecting dual redundant controllers to the host

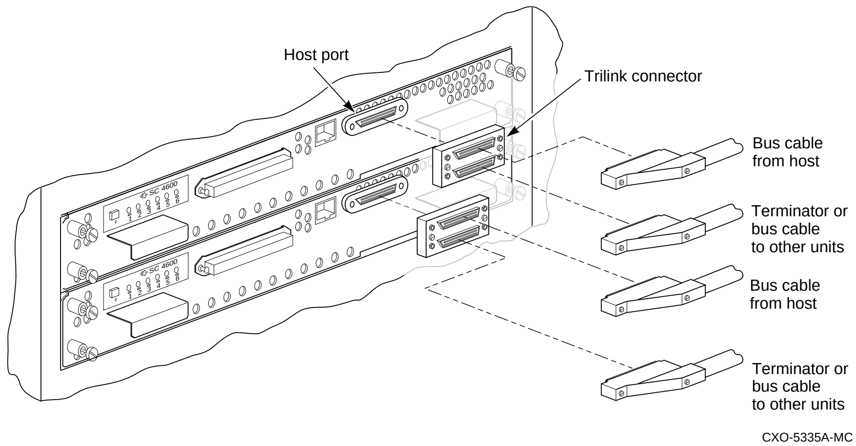


To connect a pair of dual-redundant controllers to the host:

1. Stop all I/O from the host to its devices on the bus to which you are connecting the controllers.
2. Remove the trilink connectors (12-39921-01) from both controllers. These connectors are 68-pin Y-adapters that maintain bus continuity even when they are disconnected from their controller.
3. Connect the bus cable from the host to one of the connectors on the front of one of the trilink connectors as shown in Figure 2—3.
4. Connect the two trilink connectors with a jumper cable.
5. If the controllers are at the end of the bus, connect a terminator to the open connector on the front of the trilink connector. Otherwise, connect a cable that continues to the next device on the bus. (Be sure to install a terminator at the end of the bus.)
6. Reconnect the trilink connectors to the host ports on the controllers. Do not disconnect the host cables from the trilink connector.
7. Route and tie the cables as desired.
8. Restart the I/O from the host. Some operating systems may require you to reboot the host to see the devices attached to the new controller.

Connecting multiple bus failover, dual-redundant controllers to the host

Figure 2—4 Connecting multiple bus failover, dual redundant controllers to the host



To connect a pair of multiple bus failover dual-redundant controllers to the host:

1. Stop all I/O from the host to its devices on the bus to which you are connecting the controllers.
2. Remove the trilink connectors (12-39921-01) from both controllers. These connectors are 68-pin Y-adapters that maintain bus continuity even when they are disconnected from their controller.
3. Connect a bus cable from the host to one of the connectors on the front of each trilink connector as shown in Figure 2—4.
4. If the controllers are at the end of the bus, connect a terminator to the open connectors on the front of each trilink connector. Otherwise, connect a cable that continues to the next device on each bus. (Be sure to install a terminator at the end of the bus.)
5. Reconnect the trilink connectors to host ports on the controllers. Do not disconnect the host cables from the trilink connectors.

6. Route and tie the cables as desired.
7. Restart the I/O from the host. Some operating systems may require you to reboot the host to see the devices attached to the new controller.

3

Planning your storagesets

Creating a profile

Creating a profile for your storagesets and devices will greatly simplify the configuration process. This chapter helps you to choose the storagesets that best suit your needs and to make informed decisions about the switches that you can enable for each storageset or storage device that you configure in your subsystem.

Take a few moments to familiarize yourself with the kinds of information contained in a storageset profile.

Figure 3—1 A typical storageset profile

Storageset profile

Type

☒ RAIDset ☐ Mirrorset ☐ Stripeset ☐ Striped mirrorset

Storageset name accept CFMENU default=

Disk drives DISK130, DISK230, DISK330

Unit number accept CFMENU default=

Partitions

Unit #	Unit #	Unit #	Unit #	Unit #	Unit #	Unit #	Unit #
%	%	%	%	%	%	%	%

RAIDset switches

Reconstruction policy
☒ Normal (default)
☐ Fast

Reduced membership
☒ No (default)
☐ Yes, missing:

Replacement policy
☒ Best performance (default)
☐ Best fit
☐ None

Mirrorset switches

Replacement policy
☐ Best performance (default)
☐ Best fit
☐ None

Copy policy
☐ Normal (default)
☐ Fast

Read source
☐ Least busy (default)
☐ Round robin
☐ Disk drive:

Initialize switches

Chunksize
☒ Automatic (default)
☐ 64 blocks
☐ 128 blocks
☐ 256 blocks
☐ Other:

Metadata
☒ Destroy (default)
☐ Retain

Saved configuration
☐ No (default)
☒ Yes

UNIT switches

Read cache
☒ Yes (default)
☐ No

Write cache
☐ No (default)
☒ Yes

Maximum cache transfer
☒ 32 blocks (default)
☐ Other:

Availability
☒ Run (default)
☐ NoRun

Write protection
☒ No (default)
☐ Yes

The appendix contains blank profiles that you can copy and use to record the details for your storagesets.

Defining your storage requirements

Start the planning process by defining your storage requirements. Here are a few of the questions you should ask yourself:

- What applications or user groups will access the subsystem? How much capacity do they need?
- What are the I/O requirements? If an application is data-transfer intensive, what is the required transfer rate? If it is I/O-request intensive, what is the required response time? What is the read/write ratio for a typical request?
- Are most I/O requests directed to a small percentage of the disk drives? Do you want to keep it that way or balance the I/O load?
- Do you store mission-critical data? Is availability the highest priority or would standard backup procedures suffice?

Choosing a storageset

Different applications may have different storage requirements, so you'll probably want to configure more than one kind of storageset in your subsystem.

All of the storagesets described in this book are implementations of RAID, or Redundant Array of Independent Disks. Consequently, they all share one important feature: each storageset, whether it contains two disk drives or ten, looks like one large, virtual disk drive to the host.

Compare the different kinds of storagesets to determine which ones satisfy your requirements.

Table 3—1 A comparison of different kinds of storagesets

Storageset	Relative availability	Request rate (read/write)	Transfer rate (read/write)	Applications
Array of disk drives (JBOD)	Proportionate to number of disk drives.	Identical to single disk drive.	Identical to single disk drive.	
Stripeset (RAID 0)	Proportionate to number of disk drives; worse than single disk drive.	Excellent if used with large chunk size.	Excellent if used with small chunk size.	Applications that require high performance for non-critical data.
Mirrorset (RAID 1)	Excellent	Good/Fair	Good/Fair	System drives; critical files.
RAIDset (RAID 3/5)	Excellent	Excellent/Fair	Good/Poor	High request rates, read-intensive, data lookup.
Striped Mirrorset (RAID 0+1)	Excellent	Excellent if used with large chunk size.	Excellent if used with small chunk size.	Any critical response-time application.

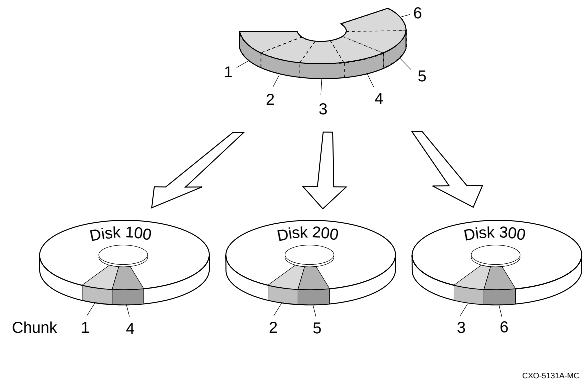
For a comprehensive discussion of RAID, refer to *The RAIDBOOK—A Source Book for Disk Array Technology,* published by the RAID Advisory Board, St. Peter, MN.

Using stripesets to increase I/O performance

Stripesets enhance I/O performance by spreading the data across multiple disk drives. Each I/O request is broken into small segments called "chunks." These chunks are then "striped" across the disk drives in the storageset, thereby allowing several disk drives to participate in one I/O request to handle several I/O requests simultaneously.

For example, in a three-member stripeset that contains disk drives 100, 200, and 300, the first chunk of an I/O request is written to 100, the second to 200, the third to 300, the fourth to 100, and so forth until all of the data has been saved.

Figure 3—2 Striping lets several disk drives participate in each I/O request



The relationship between the chunk size and the average request size determines if striping maximizes the request rate or the data-transfer rate. You can set the chunk size or let the controller set it automatically. See *Chunk size* on page 3–18 for information about setting the chunk size.

An incidental benefit of striping is that it balances the I/O load across all of the disk drives in the storageset. This can increase the subsystem's performance by eliminating the hot spots, or high localities of reference, that occur when frequently accessed data becomes concentrated on a single disk drive.

Considerations for planning a stripeset

Keep the following points in mind as you plan your stripesets:

- A storageset should only contain disk drives of the same capacity. The controller limits the capacity of each member to the capacity of the smallest member in the storageset. Thus, if you combine 2 GB disk drives with 1 GB disk drives in the same storageset, you'll waste 1 GB of capacity on each 2 GB member.

Note

If you need high performance and high availability, consider using a RAIDset, striped mirrorset, or a host-based shadow of a stripeset.

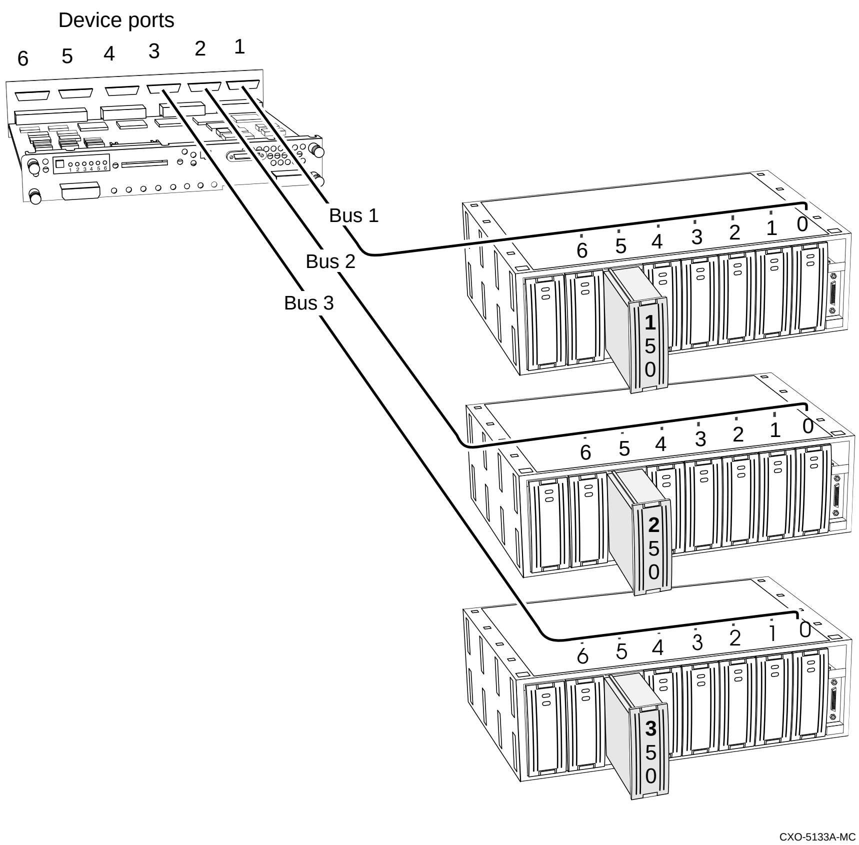
- Striping doesn’t protect against data loss. In fact, because the failure of one member is equivalent to the failure of the entire stripeset, the likelihood of losing data is higher for a stripeset than for a single disk drive.

  For example, if the mean time between failures (MTBF) for a single disk is λ hours, then the MTBF for a stripeset that comprises N such disks is λ/N hours. For example, if a single disk’s MTBF is 150,000 hours (about 17 years), a stripeset comprising four of these disks would only have an MTBF of slightly more than four years.

  For this reason, you should avoid using a stripeset to store critical data. Stripesets are more suitable for storing data that can be reproduced easily or whose loss doesn’t prevent the system from supporting its critical mission.

- Evenly distribute the members across the device ports to balance load and provide multiple paths as shown in Figure 3—3.

Figure 3—3 Distribute members across ports

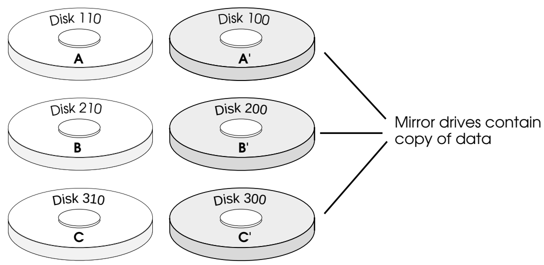


By spreading the traffic evenly across the buses, you'll ensure that no bus handles the majority of data to the storageset.

Using mirrorsets to ensure availability

Mirrorsets use redundancy to ensure availability. For each primary disk drive, there is at least one mirror disk drive. Thus, if a primary disk drive fails, its mirror drive immediately provides an exact copy of the data.

Figure 3—4 Mirrorsets maintain two copies of the same data



Considerations for planning a mirrorset

Keep these points in mind as you plan your mirrorsets:

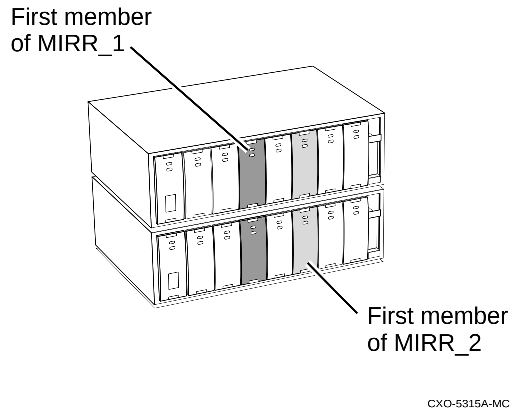
Note

If availability is your top priority, consider using redundant power supplies and dual-redundant controllers.

- Data availability with a mirrorset is excellent but costly —you'll need twice as many disk drives to satisfy a given capacity requirement.
- You can configure up to 20 mirrorsets per controller or pair of dual-redundant controllers. Each mirrorset may contain up to six members.
- A write-back cache module is required for mirrorsets.
- If you're using more than one mirrorset in your subsystem, you should put the first member of each mirrorset on different busses. (The first member of a mirrorset is the first disk drive you add with CFMENU or the ADD MIRRORSET command.)

  When a controller receives a request to read or write data to a mirrorset, it typically accesses the first member of the mirrorset. If you have several mirrorsets in your subsystem and their first members are on the same bus, that bus will be forced to handle the majority of traffic to your mirrorsets.

Figure 3—5 Put first members on different busses



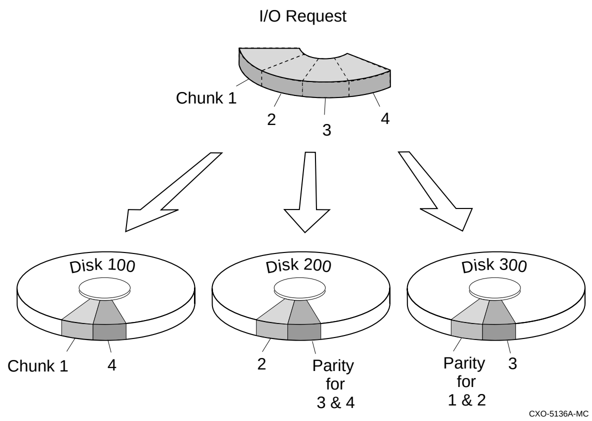
To avoid an I/O bottleneck on one bus, you can simply put the first members on different busses—in other words, put them in different shelves, as shown in Figure 3—5. Additionally, you can set the read-source switch to Round Robin. See *Read source* on page 3–17 for more information about this switch.

- A storageset should only contain disk drives of the same capacity. The controller limits the capacity of each member to the capacity of the smallest member in the storageset. Thus, if you combine 2 GB disk drives with 1 GB disk drives in the same storageset, you'll waste 1 GB of capacity on each 2 GB member.
- Evenly distribute the members across the device ports to balance load and provide multiple paths as shown in Figure 3—3.

Using RAIDsets to increase performance and availability

RAIDsets are enhanced stripesets—they use striping to increase I/O performance and distributed-parity data to ensure data availability.

Figure 3—6 Parity ensures availability; striping provides good performance



Just as with stripeset, the I/O requests are broken into smaller "chunks" and striped across the disk drives until the request is read or written. But, in addition to the I/O data, chunks of parity data —derived mathematically from the I/O data—are also striped across the disk drives. These parity data enable the controller to reconstruct the I/O data if a disk drive fails. Thus, it becomes possible to lose a disk drive without losing access to the data it contained. (Data could be lost if a second disk drive fails before the controller replaces the first failed disk drive.)

For example, in a three-member RAIDset that contains disk drives 100, 200, and 300, the first chunk of an I/O request is written to 100, the second to 200, then parity is calculated and written to 300; the third chunk is written to 300, the forth to 100, and so forth until all of the data is saved.

The relationship between the chunk size and the average request size determines if striping maximizes the request rate or the data-transfer rates. You can set the chunk size or let the controller set it automatically. See *Chunk size* on page 3–18 for information about setting the chunk size.

Considerations for planning a RAIDset

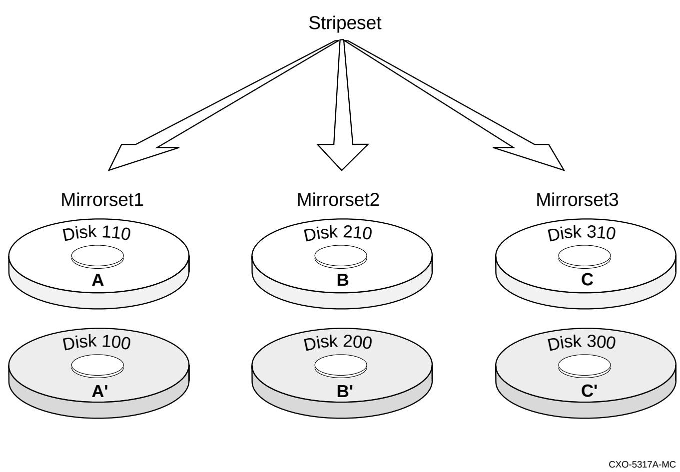
Keep these points in mind as you plan your RAIDsets:

- A write-back cache module is required for RAIDsets, but write-back cache needn't be enabled for the RAIDset to function properly.
- A RAIDset must include at least three disk drives, but no more than 14.
- Evenly distribute the members across the device ports to balance load and provide multiple paths as shown in Figure 3—3.
- A storageset should only contain disk drives of the same capacity. The controller limits the capacity of each member to the capacity of the smallest member in the storageset. Thus, if you combine 2 GB disk drives with 1 GB disk drives in the same storageset, you'll waste 1 GB of capacity on each 2 GB member.

Using striped mirrorsets for the highest performance and availability

Striped mirrorsets are simply stripesets whose members are mirrorsets. Consequently, this kind of storageset combines the performance of striping with the reliability of mirroring. The result is a storageset with very high I/O performance and high data availability.

Figure 3—7 Striping and mirroring in the same storageset



The failure of a single disk drive has no effect on this storageset's ability to deliver data to the host and, under normal circumstances, it has very little effect on performance. Because striped mirrorsets don't require any more disk drives than mirrorsets, this storageset is an excellent choice for data that warrants mirroring.

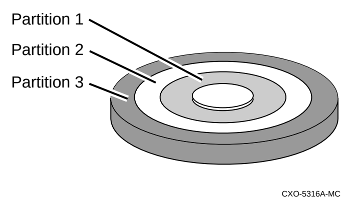
Considerations for planning a striped mirrorset

Plan the mirrorset members, then plan the stripeset that will contain them. Follow the considerations for stripesets and mirrorsets provided in this chapter.

Planning your partitions

Use partitions to divide a storageset or disk drive into smaller pieces, each of which can be presented to the host as its own storage unit. Figure 3—8 shows the conceptual effects of partitioning a single-disk unit.

Figure 3—8 Each partition can be presented to the host as a storage unit



You can create up to four partitions per disk drive, RAIDset, mirrorset, stripeset, or striped mirrorset. Each partition has its own unit number so that the host can send I/O requests to the partition just as it would to any unpartitioned storageset or device. Because partitions are separately addressable storage units, you can partition a single storageset to service more than one user group or application.

Defining a partition

Partitions are expressed as a percentage of the storageset or single disk unit that contains them. For mirrorsets and single disk units, the controller allocates the largest whole number of blocks that are equal to or less than the percentage you specify. For RAIDsets and stripesets, the controller allocates the largest whole number of stripes that are less than or equal to the percentage you specify. For stripesets, the stripe size = chunk size * number of members. For RAIDsets, the stripe size = chunk size * (number of members-1).

In any case, an unpartitioned storage unit has more capacity than a partition that uses the whole unit. That's because each partition requires 5 blocks of administrative metadata. Thus, a single disk unit that contains one partition can store *n*-5 blocks of user or application data.

See *Partitioning a storageset or disk drive* in Chapter 5 for information on manually partitioning a storageset or single-disk unit.

Guidelines for partitioning storagesets and disk drives

Keep these points in mind as you plan your partitions:

- You can create up to four partitions per storageset or disk drive.
- All of the partitions on the same storageset or disk drive must be addressed through the same controller. Thus, if you set a preferred controller for one partition in a storageset, all of the partitions in that storageset will inherit that preferred controller. This ensures a transparent failover of devices should one of the dual -redundant controllers fail.
- Partitions cannot be combined into storagesets. For example, you can't divide a disk drive into three partitions, then combine those partitions into a RAIDset.
- Partitioned storagesets and single-disk units cannot function in multiple bus failover dual-redundant configurations. For this reason, you'll have to delete your partitions before configuring the controllers for multiple busfailover.
- Once you've partitioned a container, you cannot unpartition it without reinitializing the container.
- Just as with storagesets, you don't have to assign unit numbers to partitions until you're ready to use them.

Choosing switches for your storagesets and devices

Depending upon the kind of storageset or device you're configuring, you can enable the following kinds of options or "switches:"

- RAIDset and Mirrorset switches
- Initialize switches
- Unit switches
- Device switches

Enabling switches

If you use CFMENU to configure the device or storageset, it prompts you for the switches during the configuration process and automatically applies them to the storageset or device.

If you use CLI commands to configure the storageset or device manually, the procedures in Chapter 5 indicate when and how to enable each switch.

Changing switches

You can change the RAIDset, Mirrorset, Device, and Unit switches at any time. See *Changing switches for a storageset or device* in Chapter 5.

You can't change the Initialize switches without destroying the data on the storageset or device. These switches are integral to the formatting and can only be changed by re-initializing the storageset. (Initializing a storageset is similar to formatting a disk drive; all of the data is destroyed during this procedure.)

RAIDset switches

You can enable the following kinds of switches to control how a RAIDset behaves to ensure data availability:

- Replacement policy
- Reconstruction policy
- Membership

Replacement policy

Specify a replacement policy to determine how the controller replaces a failed disk drive:

- POLICY=BEST_PERFORMANCE (default) puts the failed disk drive in the failedset then tries to find a replacement (from the spareset) that is on a different device port than the remaining, operational disk drives. If more than one disk drive meets this criterion, this switch selects the drive that also provides the best fit.

- POLICY=BEST_FIT puts the failed disk drive in the failedset then tries to find a replacement (from the spareset) that most closely matches the size of the remaining, operational disk drives. If more than one disk drive meets this criterion, this switch selects the one that also provides the best performance.
- NOPOLICY puts the failed disk drive in the failedset and doesn’t replace it. The storageset operates with less than the nominal number of members until you specify a replacement policy or manually replace the failed disk drive.

Reconstruction policy

Specify the speed with which the controller reconstructs the data from a failed disk drive then writes it to a replacement disk drive:

- RECONSTRUCT=NORMAL (default) balances the overall performance of the subsystem against the need for reconstructing the replacement disk drive.
- RECONSTRUCT=FAST gives more resources to reconstructing the replacement disk drive, which may reduce the subsystem’s overall performance during the reconstruction task.

Membership

Indicate to the controller that the RAIDset you’re adding is complete or “reduced,” which means it’s missing one of its members:

- NOREDUCED (default) indicates to the controller that all of the disk drives are present for a RAIDset.
- REDUCED lets you add a RAIDset that’s missing one of its members. For example, if you dropped or destroyed a disk drive while moving a RAIDset, you could still add it to the subsystem by using this switch.

Mirrorset switches

You can enable the following switches to control how a mirrorset behaves to ensure data availability:

- Replacement policy
- Copy speed
- Read source

Replacement policy

Specify a replacement policy to determine how the controller replaces a failed disk drive:

policy=best_performance (default) puts the failed disk drive in the failedset then tries to find a replacement (from the spareset) that is on a different device port than the remaining, operational disk drives. If more than one disk drive meets this criterion, this switch selects the drive that also provides the best fit.

- POLICY=BEST_FIT puts the failed disk drive in the failedset then tries to find a replacement (from the spareset) that most closely matches the size of the remaining, operational disk drives. If more than one disk drive meets this criterion, this switch selects the one that also provides the best performance.
- nopolicy puts the failed disk drive in the failedset and doesn’t replace it. The storageset operates with less than the nominal number of members until you specify a replacement policy or manually replace the failed disk drive.

Copy speed

Specify a copy speed to determine the speed with which the controller copies the data from an operational disk drive to a replacement disk drive:

- COPY=NORMAL (default) balances the overall performance of the subsystem against the need for reconstructing the replacement disk drive.
- COPY=FAST allocates more resources to reconstructing the replacement disk drive, which may reduce the subsystem’s overall performance during the reconstruction task.

Read source

Specify the read source to determine how the controller reads data from the members of a mirrorset:

- READ_SOURCE=ROUND ROBIN (default) forces the controller to read data sequentially from all "normal" or operational members in a mirrorset. For example, in a four-member mirrorset (A, B, C, and D), the controller reads from A, then B, then C, then D, then A, then B, and so forth. No preference is given to any member.
- READ_SOURCE=LEAST BUSY forces the controller to read data from the "normal" or operational member that has the least-busy work queue.
- READ_SOURCE=DISK*nnn* forces the controller to always read data from a particular "normal" or operational member. If the specified member fails, the controller reads from the least busy member.

Device switches

When you add a disk drive or other storage device to your subsystem, you can enable the following switches:

- Transportability
- Transfer rate

Transportability

Note

TRANSPORTABLE is especially useful for moving a disk drive from a workstation into your StorageWorks subsystem. When you add a disk drive as transportable, you can configure it as a single-disk unit and access the data that was previously saved on it.

Indicate whether a disk drive is transportable or not when you add it to your subsystem:

- NOTRANSPORTABLE disk drives (default) are marked with StorageWorks-exclusive metadata. This metadata supports the error-detection and recovery methods that the controller uses to ensure data availability. Disk drives that contain this metadata can't be used in non-StorageWorks subsystems.

- TRANSPORTABLE disk drives can be used in non-StorageWorks subsystems. Transportable disk drives can be used as single-disk units in StorageWorks subsystem as well as disk drives in other systems. They can't be combined into storagesets in a StorageWorks subsystem.

Transfer rate

Specify a transfer rate that the controller uses to communicate with the device. Use one of these switches to limit the transfer rate to accommodate long cables between the controller and a device, such as a tape library. Use one of the following values:

- TRANSFER_RATE_REQUESTED=10MHZ (default)
- TRANSFER_RATE_REQUESTED=5MHZ
- TRANSFER_RATE_REQUESTED= ASYNCHRONOUS

Initialize switches

Note

After you've initialized the storageset or disk drive, you cannot change these switches without reinitializing the storageset or disk drive.

You can enable the following kinds of switches to affect the format of a disk drive or storageset:

- Chunk size (for stripesets and RAIDsets only)
- Save configuration
- Overwrite

Chunk size

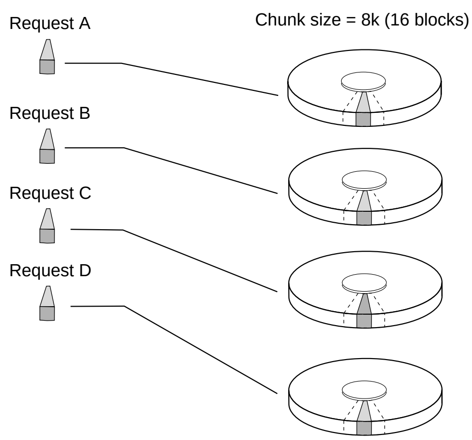
Specify a chunk size to control the stripesize used for RAIDsets and stripesets:

- CHUNKSIZE=DEFAULT lets the controller set the chunk size based on the number of disk drives (d) in a stripeset or RAIDset. If $d \leq 9$ then chunk size = 256. If $d > 9$ then chunk size = 128. However, if the cache size < 16MB then chunk size = 64 regardless of d.
- CHUNKSIZE=*n* lets you specify a chunk size in blocks. The relationship between chunk size and request size determines whether striping increases the request rate or the data -transfer rate.

Increasing the request rate

A large chunk size (relative to the average request size) increases the request rate by allowing multiple disk drives to respond to multiple requests. If one disk drive contains all of the data for one request, then the other disk drives in the storageset are available to handle other requests. Thus, in principle, separate I/O requests can be handled in parallel, thereby increasing the request rate.

Figure 3—9 If chunk size is larger than the request size, then each disk drive in the storageset can respond to a separate I/O request.



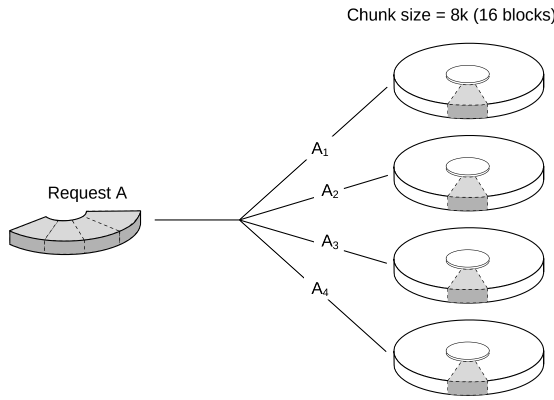
Applications such as interactive transaction processing, office automation, and file services for general timesharing tend to require high I/O request rates.

Large chunk sizes also tend to increase the performance of random reads and writes. DIGITAL recommends a chunk size of 10 to 20 times the average request size, rounded up to the nearest multiple of 64. In general, a chunk size of 256 works well for UNIX systems; 128 works well for OpenVMS systems.

Increasing the data transfer rate

A small chunk size relative to the average request size increases the data transfer rate by allowing multiple disk drives to participate in one I/O request.

Figure 3—10 Chunk size is smaller than the request size, then more than one disk drive can respond to the same I/O request.



Applications such as CAD, image processing, data collection and reduction, and sequential file processing tend to require high data -transfer rates.

Increasing sequential write performance

For stripesets (or striped mirrorsets), use a large chunk size relative to the I/O size to increase the sequential write performance. A chunk size of 256 generally works well.

Chunk size doesn’t significantly affect sequential read performance.

Maximum chunk size for RAIDsets

Don’t exceed the following chunk sizes for a RAIDset. (The maximum chunk size is derived by 2048/(d – 1) where d is the number of disk drives in the RAIDset.)

Table 3—2 Maximum chunk sizes for a RAIDset

RAIDset size	Max chunk size
3 members	1024 blocks
4 members	682 blocks

RAIDset size	Max chunk size
9 members	256 blocks
10 members	227 blocks

RAIDset size	Max chunk size
5 members	512 blocks
6 members	409 blocks
7 members	341 blocks
8 members	292 blocks

RAIDset size	Max chunk size
11 members	204 blocks
12 members	186 blocks
13 members	170 blocks
14 members	157 blocks

Save configuration

Indicate whether or not to save the subsystem's configuration on the storage unit when you initialize it:

- NOSAVE_CONFIGURATION (default) means that the controller stores the subsystem's configuration in its nonvolatile memory. Although this is generally secure, the configuration could be jeopardized if the controller fails. For this reason, you should initialize at least one of your storage units with the SAVE_CONFIGURATION switch enabled.
- SAVE_CONFIGURATION only requires one disk be initialized with this option, but more may be used, if desired. DIGITAL *does not* recommend initializing *all* disks with the SAVE_CONFIGURATION switch, because every update to non-volatile memory causes writes to all disks and can affect performance adversely.
- Specify SAVE_CONFIGURATION when initializing any disk device or container on which you want to store a copy of the controller configuration. If you use the switch for a multi-device storageset, such as a stripeset, the complete information is stored on each device in the storageset.
- DIGITAL recommends that the SAVE_CONFIGURATION switch only be used for single controller configurations. (Use the SET FAILOVER COPY= command to save configuration information for dual-redundant configurations).
- SAVE_CONFIGURATION allows the controller to use 256K of each device in a storage unit to save the subsystem's configuration. The controller saves the configuration when you change the configuration or add a patch to your controller. If the controller should fail, you can recover your latest configuration from the storage unit rather than rebuild it from scratch.
- Specify NOSAVE_CONFIGURATION for devices and storagesets on which you do not want to store a copy of the controller configuration.

- SAVE_CONFIGURATION is not available for upgrades of firmware or hardware, and will not perform inter-platform conversions.

Overwrite

Specify whether to destroy or retain the user data and metadata when you're initializing a disk drive that has been previously used in a storageset or as a single-disk unit:

- DESTROY (default) overwrites the user data and forced-error metadata on a disk drive when it's initialized.

Note

NODESTROY is ignored for members of a RAIDset, all of which are destroyed when the RAIDset is initialized.

- NODESTROY preserves the user data and forced-error metadata when a disk drive is initialized. Use NODESTROY to create a single-disk unit from any disk drive that has been used as a member of a mirrorset. See the *Reduced* command in the *CLI Reference Manual* for information on removing disk drives from a mirrorset.

Unit switches

You can enable the following Unit switches for the following storagesets and devices

Table 3—3 Unit switches

Unit switch	RAID	Stripe	Mirror	Disk Notrans	Disk Trans	CDROM	Tape	Pass through
Preferred path for multiple bus failover configurations	✓	✓	✓	✓	✓	✓	✓	✓
Read cache	✓	✓	✓	✓	✓	✓		
Writeback cache	✓	✓	✓	✓		✓		
Maximum cache Transfer	✓	✓	✓	✓	✓	✓		
Availability	✓	✓	✓	✓	✓	✓		
Tape format							✓	
Write protection	✓	✓	✓	✓	✓			✓

Preferred path for multiple bus failover configurations

Note

The PREFERRED PATH switch applies only to storage units in a multiple bus failover configuration. See page 3–26 to find out how you can use unit numbers to establish preferred paths for storage units in a dual-redundant configuration.

Specify which controller accesses the storage unit. If one controller fails, the operational controller will handle the I/O activity to all of the storage units regardless of their preferred paths:

- NOPREFERRED_PATH (default) allows either controller to access the storage unit.
- PREFERRED_PATH=THIS_CONTROLLER indicates that the controller to which you're connected handles all I/O activity to the storage unit. By establishing preferred paths, you can distribute the I/O load evenly between the two controllers by dividing the storage units into two equal groups—based on their I/O activities—and assigning each group to its own controller.
- PREFERRED_PATH=OTHER_CONTROLLER indicates that the other controller—the one to which you're not connected—handles all I/O activity to the storage unit.

Read cache

Enable or disable the caching of read data to the storage unit:

- READ_CACHE (default) enables the caching of read data.
- NOREAD_CACHE disables the caching of read data.

Write-back cache

Note

If you disable write-back caching for a storage unit that previously used it, it may take up to five minutes to flush the unwritten data from the cache to the devices in the storage unit. Use the SHOW *unit-name* command to check whether the cache is flushed.

Enable or disable the controller's write-back caching for a storage unit:

- WRITEBACK_CACHE enables write-back caching.
- NOWRITEBACK_CACHE disables write-back caching.

Considerations When Using Write-back Caching

The following list summarizes considerations you must be aware of when using write-back cache.

Caution

Two conditions will cause data contained within write-back cache to be lost: if power from the main power supply and the external cache battery is interrupted, or if the cache module is removed before the controller flushes the data from the write-back

- When restarted, the controller attempts to flush any unflushed data within write-back cache to the devices. However, by specifying the IGNORE_ERRORS or IMMEDIATE_SHUTDOWN switch, you allow data to reside in write-back cache when the controller is turned off, regardless of any errors detected.
- RAIDsets and mirrorsets require data to be stored in write-back cache to accomodate the write hole and to increase performance —without regard to the WRITEBACK_CACHE switch setting.
- If data is contained within the write-back cache while the subsystem is shut down, do not perform any hardware changes until after the controller flushes the data to the devices.
- When restarted, the controller attempts to flush any unflushed data within the write-back cache to the devices. However, by specifying the

IGNORE_ERRORS or IMMEDIATE_SHUTDOWN switch, you allow data to reside in write-back cache when the controller is turned off, regardless of any errors detected.

Maximum cache transfer

Specify the amount of data (in blocks) that the controller may cache to satisfy a read request:

- MAXIMUM_CACHED_TRANSFER=*n* lets you indicate the number of data blocks that the controller will cache to satisfy a read request. Any I/O transfers in excess of the specified size will not be cached. You can specify a value from 1 to 1024.
- MAXIMUM_CACHED_TRANSFER=32 (default) is the default number of data blocks that the controller will cache to satisfy a read request.

Availability

Specify whether or not to make the storage unit available to the host:

- RUN (default) specifies that as soon as you provide a host-addressable unit number the storage unit will be made available to the host.
- NORUN specifies that the storage unit will not be made available to the host until you issue a SET *unit-number* RUN command..

Tape format

Specify the tape format to be used unless it's overridden by the host. Not all tape drives support all formats:

- DEFAULT_FORMAT=DEVICE_DEFAULT lets the controller automatically determine and set the device default for the tape drive.
- DEFAULT_FORMAT=*n* lets you specify the tape format, such as TZ88 or HOST_SELECTED.

  To display the formats that are supported, enter the following command at the CLI prompt:

  ```
  CLI> SHOW tape-unit-number DEFAULT_FORMAT= ?
  ```

Write protection

Enable or disable write protection for the storage unit:

- NOWRITE_PROTECT (default) enables the controller to write new data to the storage unit.
- WRITE_PROTECT prevents the controller from writing any new data to the storage unit. (The controller can write to a protected unit if it needs to reconstruct data.)

Assigning unit numbers

A controller can respond to four SCSI target IDs, each of which can present up to eight logical unit numbers (LUNs) to a host. This means that each controller or dual-redundant pair of controllers can present up to 32 storage units to a host.

You'll need to assign a unique unit number to each storageset, single disk unit, or storage device that you want your host to know about in your subsystem. A unit number is an alpha-numeric tag that identifies each storage unit in your subsystem, such as D102 for a disk-based storage unit. The host uses these numbers to indicate the source or target for every I/O request it sends to a controller.

Each four-place unit number contains the following:

- A letter that indicates the kind of devices in the storage unit: use D for disk drives (including CD-ROMs) or P for passthrough devices for tape drives, loaders, and libraries. (If you're using CFMENU to configure your storagesets and devices, it will automatically supply a device letter for you.)

Note

By carefully choosing the first number, you can establish preferred paths for all of your storage units in a dual-redundant configuration.

- A first number that indicates which controller accesses the storage unit during normal operation. Use one of the controller's SCSI target IDs (0-7). Omit the leading zeroes for storage units associated with the controller's SCSI target ID zero. For example, use D2 instead of D002 for a storageset that's accessed through the controller's SCSI target ID 0.
- A second number that is always zero.
- A third number that identifies the logical unit number (LUN) for the device or storage unit (0-7). This number is often called the "storageset ID."

Creating a storageset map

Configuring your subsystem will be easier if you know how the storagesets correspond to the disk drives in your subsystem. You can see this relationship by creating a storageset map like the one shown here.

Table 3—4 A storageset map for a subsystem that contains two RAIDsets, two mirrorsets, and four disk drives in the spareset. Each shelf also has dual power supplies.

SW800 and SW500 cabinets

Note

This map provides more than enough PTL slots to map a controller or a pair of dual-redundant controllers. A single controller can support up to 42 devices. Dual-redundant controllers can support up to 36 devices. Each shelf can support an optional power supply in addition to the ones shown.

Power supply	PWR 2		M1	R2	R2	R1	R1
Power supply	PWR 2		M1	R2	R2	R1	R1
Power supply	PWR 2		M2	R2	R2	R1	R1
Power supply	PWR 2		M2	Spare	Spare	Spare	Spare
Power supply							
Power supply							
Power supply							

To create a storageset map:

1. Copy the appropriate cabinet template from the Appendix.
2. Establish a local or remote connection to one of the controllers in your subsystem.
3. Show the devices that are assigned to the controller:

   CLI> **SHOW DEVICES**

4. Locate each device assigned to the controller and record its location on your copy of the cabinet template:

   CLI> **LOCATE *device_name***

   The LOCATE command causes the device's LED to flash continuously. To turn off the LED:

   CLI> **LOCATE CANCEL**

   The controller names each device based on its Port -Target-LUN (PTL) location. See *PTL addressing convention* below).

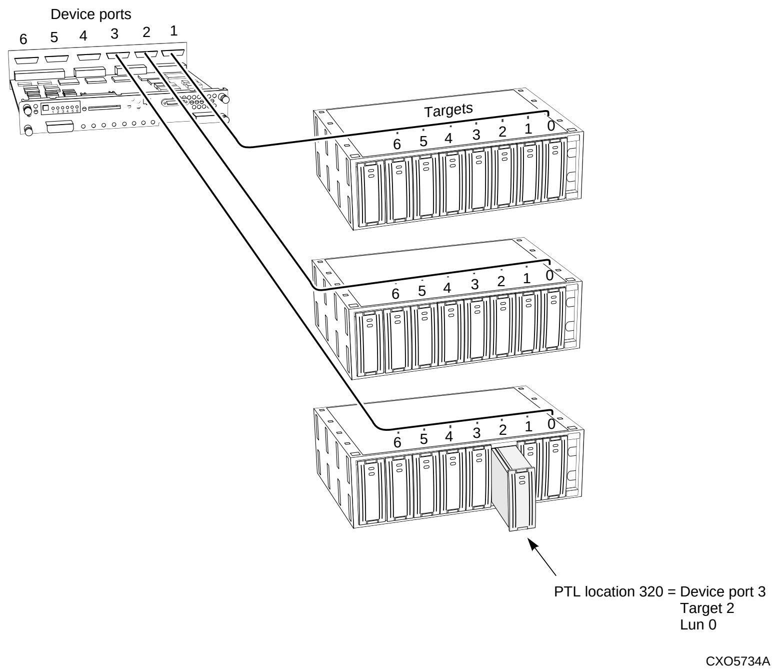
5. Repeat steps 2 through 4 for each controller or dual -redundant pair of controllers.
6. After you have mapped the devices to your cabinet template, create the storageset map by circling each group of disk drives that you want to combine into a storageset or put into the spareset. Label each group with its storageset name, for example: RAID1 for a RAIDset; Mirr1 for a mirrorset; and Stripe1 for a stripeset.

PTL addressing convention

Your controller has six SCSI–2 device ports. Each device port connects to a shelf that supports up to seven devices or "targets." And every device uses LUN 0, except some tape loaders, which use LUN 1.

As shown in Figure 3—11, the controller addresses DISK320 through device port 3, target 2, LUN 0. Thus, the PTL location indicates the pathway that the controller uses to address a disk or tape drive. It also indicates the device name.

Figure 3—11 PTL addressing



The controller uses the PTL location to name each device that you add to your subsystem with the CONFIG utility or CFMENU. (Factory-installed devices are added with the CONFIG utility. Thus, their names derive from their PTL locations.) For example, if the controller finds a disk in PTL 320, it names it DISK320; if it finds a tape drive at PTL 320, it names it TAPE320.

When your controller receives an I/O request, it identifies the unit number for the request, then correlates the unit number to the storageset name. From the storageset name, the controller locates the appropriate devices for the I/O request. (For example, the RAIDset "R1" might contain DISK150, DISK250, and DISK350.) The controller generates the read or write request to the appropriate device using the PTL addressing convention.

The next step...

Turn to Chapter 4 if you want to configure your storage units automatically with CFMENU, a menu-based utility that simplifies the configuration process. This utility is especially helpful if you're configuring storagesets for the first time.

Turn to Chapter 5 if you want to configure your storage units manually by issuing CLI commands from a local or remote connection. Configuring your storage units manually give you more authority when it comes to naming the storage units, since CFMENU automatically names them for you. However, with authority comes responsibility, so you should complete a profile for each storage unit or device that you want configure in your subsystem before you begin the procedures provided in Chapter 5.

4

Automatically configuring storagesets

Introducing CFMENU

CFMENU is a modest but effective utility that simplifies the task of configuring storagesets and devices.

Note

See Chapter 5 if you want to modify an existing storageset or configure a tape drive or tape loader.

With CFMENU you are free to think about what you want to do rather than how to get the controller to do it. Instead of issuing CLI commands, you choose from a menu of configuration tasks, such as adding a storageset or assigning a unit number.

Based on your choices, CFMENU prompts you for the information it needs to complete the task. It even prompts you to specify the switches you want to enable for a storageset or device.

CFMENU uses columns of information to let you know what is going on during the configuration process. These columns are displayed on the Main Menu and other sub-menus and are continually updated to reflect the current configuration.

Table 4–1 lists the heading and contents for each column on the Main Menu.

Table 4–1 Interpreting CFMENU Columns

Column heading	Information Displayed
Main menu	Shows the tasks you can accomplish with CFMENU.
Unconfig’d Dev.PTLs	Shows the PTL locations of devices that have not yet been added to the controller’s configuration. Use these devices to create single-disk units and storagesets, such as stripesets and RAIDsets.
Config’d PTLs	Shows the PTL locations of all devices that are used in—or are eligible to be used in—a storageset or a single-disk unit.
Device Name	Shows the names of all devices that are used in—or are eligible to be used in—a storageset or as a single-disk unit.
Product ID	Shows the model numbers of all devices that are used in—or are eligible to be used in—a storageset or as a single-disk unit.
Stor.set Name	Shows the name of all storagesets in the controller’s list of configured storagesets: by convention, Sn for stripesets, Mn for mirrorsets, and Rn for RAIDsets.

Column heading	**Information Displayed**
Stor.set Typ/Sz	Shows the types of storagesets and their number of members. For example, STR/5 is a stripeset that contains five disk drives; MIR/2 is a mirrorset that contains two disk drives.
Chunk Size	Shows the chunk size for stripesets and RAIDsets. This column is marked “unk” (unknown) until you initialize the storageset.
Trnsp.	Displays “Y” if you enabled the Transportable switch.
Init’d	Displays “Y” if you initialized the storageset.
Reduc	Displays “Y” if you initialized the storageset with the Reduced switch or if the storageset is in a reduced state due to the failure of one of its members.
Unit	Shows the unit numbers for all storagesets or devices.
PT	Displays “P” if the unit is partitioned.
WP	Displays “Y” if the unit is write protected.
WB	Displays “Y” if you enabled write-back cache.

Considerations for using CFMENU

Keep the following points in mind for using CFMENU:

- Configure your storagesets manually if you want to use your own naming scheme. CFMENU names each storageset according to a simple naming convention: M*n* for mirrorsets, S*n* for stripesets, and R*n* for RAIDsets, where *n* is a sequentially indexed number. CFMENU also automatically provides unit-number prefixes; you specify the actual number.
- You can create and delete storagesets with CFMENU, however, you cannot modify them once they have been created. Follow the steps in *Changing switches for a storageset or device* in Chapter 5.
- If you are using dual-redundant controllers, you do not need to run CFMENU on both controllers simultaneously. The "other controller" automatically inherits the configuration you create with CFMENU.
- CFMENU cannot configure tape loaders. See *Configuring a tape drive* and *Configuring a tape loader* in Chapter 5.

Adding devices with CFMENU

To add a disk drive or other storage device to your subsystem with CFMENU:

1. Install the new disk drives in your storage cabinet.
2. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

3. From the Main Menu, choose task 1 to go to the Device Menu.
4. From the Device Menu, choose task 1 to add disk drives.
5. CFMENU presents—one at a time—the disk drives or devices that you may add to the subsystem. Type Y to add the disk drive, N to skip to the next one.
6. Set the disk drive NOTRANSPORTABLE. See *Initialize switches* in Chapter 3 for more information about this switch.
7. Return to the Main Menu and exit CFMENU.

Deleting Devices with CFMENU

To delete a disk drive or other storage device from your subsystem with CFMENU:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. From the Main Menu, choose task 1 to go to the Device Menu.
3. From the Device Menu, choose task 2 to delete devices.
4. CFMENU presents—one at a time—the disk drives or devices that you may delete from the subsystem. Type Y to delete the disk drive, N to skip to the next one.
5. Return to the Main Menu and exit CFMENU.

Creating a Storageset with CFMENU

Creating a storageset or single-disk unit with CFMENU is as easy as choosing menu options and responding to prompts. Just remember to move through the Main Menu items from top to bottom for each storageset or single-disk unit you want to create.

To create a storageset or single-disk unit with CFMENU:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. Go to step 9 if you are configuring a single-disk unit, otherwise choose task 2, 3, or 4 depending on the kind of storageset you want to create.
3. CFMENU displays a storageset menu that corresponds to your choice: Mirrorset Menu, Stripeset Menu, or RAIDset/sparesets/failedsets Menu.

   Note

   Press “D” to scroll down CFMENU’s columns.
   Press “U” to scroll up.

4. From the storageset menu, choose task 1 to add a storageset to the controller’s list of available storagesets.
5. Enter the number of disk drives or members that you want to include in the storageset.

6. CFMENU presents—one at a time—the disk drives or members that you may include in the storageset. Type Y to include a member, N to skip to the next one.
7. When you reach number of members specified in step 5, CFMENU prompts you for the switches you can apply to the storageset. Indicate you choice or press Return to accept the default value.
8. CFMENU displays a message that indicates the storageset's type and name, as well as the names of all its members. Press Return to create the storageset.
9. Return to the Main Menu and repeat steps 2 through 8 for each storageset or single-disk unit you want to create.
10. From the Main Menu, choose task 6 to go to the Initialization Menu.
11. From the Initialization Menu, choose task 1 to initialize the storageset (or the disk drive if you are creating a single-disk unit).
12. CFMENU prompts you for the Initialize switches you can apply to the storageset or single-disk unit. Indicate you choice or press Return to accept the default value. See *Initialize switches* in Chapter 3 for more information about these switches.
13. Repeat steps 10 and 11 for each storageset or single-disk unit you want to initialize.
14. Return to the Main Menu.
15. From the Main Menu, choose task 7 to go to the Unit Menu.
16. From the Unit Menu, choose task 1 to assign a host -addressable unit number to the storageset or single-disk unit. See *Assigning unit numbers* in Chapter 3 for more information about choosing unit numbers.
17. CFMENU prompts you for the Unit switches you can apply to the unit. Indicate your choice or press Return to accept the default value. See *Unit switches* in Chapter 3 for more information about these switches.
18. Repeat steps 15 and 16 for each storageset or single -disk unit to which you want to assign a unit number.
19. Return to the Main Menu and exit CFMENU.

Deleting a Storageset with CFMENU

To delete a storageset with CFMENU:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. From the Main Menu, choose task 2, 3, or 4 depending on the kind of storageset you want to delete.
3. From the storageset menu, choose task 2 to begin deleting storagesets.
4. CFMENU presents—one at a time—the storagesets that you may delete. Type Y to delete the storageset, N to skip to the next one.
5. Return to the Main Menu and exit.

Adding a Disk Drive to a Spareset with CFMENU

To add a disk drive to a spareset:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. From the Main Menu, choose task 4 to go to the RAIDset Menu.
3. From the RAIDset menu, choose task 4 to go to the Spareset/Failedset Menu.
4. From the Spareset/Failedset Menu, choose task 1 to add the disk drives to the spareset.
5. CFMENU presents—one at a time—the disk drives that you may add to the spareset. Type Y to add the disk drive, N to skip to the next one.
6. Return to the Main Menu and exit.

Initializing Containers with CFMENU

To initialize devices or storagesets:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. From the Main Menu, choose task 6 to go to the Initialization Menu.
3. From the Initialization Menu, choose task 1.
4. CFMENU presents—one at a time—the devices or storagesets that you may initialize. Type Y to add the disk drive, N to skip to the next one.

5. In addition, CFMENU will prompt you to decide on other operating qualifiers, depending on whether the container is a device, mirrorset, stripeset, orRAIDset. (Refer to the ADD or SET commands in the *HSZ40 Array Controllers CLI Reference Manual* if you need help understanding the qualifiers.)
6. Return to the Main Menu and exit.

Adding Units with CFMENU

To add units that are not partitioned:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. From the Main Menu, choose task 7 to go to the Unit Menu.
3. From the Unit menu, choose task 1 to add a unit.
4. CFMENU presents—one at a time—for the containers from which the new units will be created. Type Y to add the disk drive, N to skip to the next one.
5. CFMENU also prompts you to assign a unit number. (The program automatically assigns a "D" or "T" to the unit number when listing the unit.)
6. In addition, CFMENU prompts you to decide on other unit qualifiers. See the description of the ADD *unit* or SET *unit* commands in the *HSZ40 Array Controllers CLI Reference Manual*.
7. Return to the Main Menu and exit.

Deleting a Disk Drive from a Spareset with CFMENU

To delete a disk drive from a the spareset:

1. Start CFMENU:
2. **RUN CFMENU**
3. From the Main Menu, choose task 4 to go to the RAIDset Menu.
4. From the RAIDset menu, choose task 4 to go to the Spareset/Failedset Menu.
5. From the Spareset/Failedset Menu, choose task 2 to delete the disk drives from the spareset.

6. CFMENU presents—one at a time—the disk drives that you may delete from the spareset. Type Y to delete the disk drive, N to skip to the next one.

Note

CFMENU only allows you to delete one disk from a spareset at a time.

7. Return to the Main Menu and exit.

Partitioning a Container with CFMENU

To partition a storageset or single-disk container:

1. Start CFMENU:
2. CLI> **RUN CFMENU**
3. From the Main Menu, choose task 5 and answer yes to processing the partition if there is a partitioned container to go to the Partition Processing menu.
4. CFMENU presents—one at a time—the storageset or single disk containers that you may partition. Type Y to select the container, N to skip to the next one.
5. From the Partition Menu, choose task 1 to partition the selected container.
6. Indicate the percentage of the container that you want to dedicate to the first partition.
7. Repeat step 6 for each partition you want to create on the container.
8. Return to the Main Menu and exit.

Adding a Partitioned Unit with CFMENU

To add a partitioned unit to your subsystem with CFMENU:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. From the Main Menu, choose task 5 to go to the Partition Menu.
3. From the Partition Menu, choose task 2 to add partitioned units.
4. CFMENU presents—one at a time—the devices that you may add to the subsystem. Type Y to add the device, N to skip to the next one.
5. Return to the Main Menu and exit CFMENU.

Deleting a Partitioned Unit with CFMENU

To delete a partitioned unit from your subsystem with CFMENU:

1. Start CFMENU:

   ```
   CLI> RUN CFMENU
   ```

2. From the Main Menu, choose task 5 to go to the Partition Menu.
3. From the Partition Menu, choose task 3 to delete the partitioned unit.
4. CFMENU presents—one at a time—the devices that you may delete from the subsystem. Type Y to add the device, N to skip to the next one.
5. Return to the Main Menu and exit CFMENU.

5

Manually configuring storagesets

Adding disk drives

The factory-installed devices in your StorageWorks subsystem have already been added to the controller’s list of eligible devices. If you want to add new devices to your subsystem, you’ll have to issue one the following CLI commands before you can use them in any kind of storageset, single disk unit, or spareset.

Adding one disk drive at a time

To add one new disk drive to your controller’s list of eligible devices:

`CLI>` **`ADD DISK`** ***`DISKnnn PTL-location`***

Adding several disk drives at a time

To add several new disk drives to your controller’s list of eligible devices:

`CLI>` **`RUN CONFIG`**

Adding CD–ROM drives

To add new CD–ROM drives to your subsystem, issue the following CLI commands:

Adding one CD–ROM drive at a time

To add one new CD–ROM drive to your controller’s list of eligible devices:

`CLI>` **`ADD CDROM CDROM-`*****`name`*** *`SCSI-location`*

The *SCSI-location* parameter assigns a PTL address to the CD –ROM that is used by the controller.Refer to the CLI Manual for an explanation of the PTL numbering system.

Adding several CD–ROM drives at a time

To add several new disk drives to your controller’s list of eligible devices:

`CLI>` **`RUN CONFIG`**

Configuring a stripeset

See Chapter 3 for information about creating a profile and understanding the switches you can set for this kind of storage unit.

1. Create the stripeset by adding its name to the controller's list of storagesets and specifying the disk drives it contains:

   `CLI>` **`ADD STRIPESET`** ***`stripeset-name (DISKnnn DISKnnn…)`***

2. Initialize the stripeset. If you want to set any Initialize switches, you must do so in this step:

   `CLI>` **`INITIALIZE`** ***`stripeset-name SWITCH_VALUE`***

Table 5—1 Initialize

Initialize switch	Value and syntax
Chunk size	CHUNKSIZE=DEFAULT* CHUNKSIZE=*n*
Save configuration	NOSAVE_CONFIGURATION* SAVE_CONFIGURATION
Destroy	NODESTROY* DESTROY

3. Present the stripeset to the host by giving it a unit number the host can recognize:

   `CLI>` **`ADD UNIT`** ***`unit-number stripeset-name`***

4. Optional: set the Unit switches or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   `CLI>` **`SET`** ***`unit-number SWITCH_VALUE`***

Table 5—2 Unit switches

Unit switch	Value and syntax
Maximum cached transfer	MAXIMUM_CACHED_TRANSFER=32* MAXIMUM_CACHED_TRANSFER=*n*
Read cache	READ_CACHE* NOREAD_CACHE
Write-back cache	NOWRITEBACK_CACHE* WRITEBACK_CACHE
Availability	RUN* NORUN
Write protection	NOWRITE_PROTECT* WRITE_PROTECT

5. Verify the stripeset configuration and switches:

   CLI> **SHOW *stripeset-name***

6. Verify the unit configuration and switches:

   CLI> **SHOW *unit-number***

For example

The following example shows the commands you would use to create Stripe1, a three-member stripeset.

```
CLI> ADD STRIPESET Stripe1 disk100 disk200 disk300
CLI> INITIALIZE Stripe1 CHUNKSIZE=128
CLI> ADD UNIT D100 Stripe1
CLI> SET D100 MAXIMUM_CACHED_TRANSFER=16
CLI> SET D100 WRITEBACK_CACHE
CLI> SHOW Stripe1
CLI> SHOW D100
```

Configuring a mirrorset

See Chapter 3 for information about creating a profile and understanding the switches you can set for this kind of storage unit.

1. Create the mirrorset by adding its name to the controller's list of storagesets and specifying the disk drives it contains:

   CLI> **ADD MIRRORSET** ***mirrorset-name DISKnnn DISKnnn***

2. Optional: set the Mirrorset switches or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   CLI> **SET** ***mirrorset-name SWITCH_VALUE***

Table 5—3 Mirrorset switches

Mirrorset switch	Value and syntax
Replacement policy	POLICY=BEST_PERFORMANCE* POLICY=BEST_FIT NOPOLICY
Copy speed	COPY=NORMAL* COPY=FAST
Read source	READ_SOURCE=LEAST_BUSY* READ_SOURCE=ROUND_ROBIN READ_SOURCE=DISKNNN

3. Initialize the mirrorset. If you want to set any Initialize switches, you must do so in this step:

   CLI> **INITIALIZE** ***mirrorset-name SWITCH_VALUE***

Table 5—4 Initialize switches

Initialize switch	Value and syntax
Save configuration	NOSAVE_CONFIGURATION* SAVE_CONFIGURATION
Destroy	NODESTROY* DESTROY

4. Present the mirrorset to the host by giving it a unit number the host can recognize:

   ```
   CLI> ADD UNIT unit-number mirrorset-name
   ```

5. Optional: set the Unit switches or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   ```
   CLI> SET unit-number SWITCH_VALUE
   ```

Table 5—5 Unit switches

Unit switch	Value and syntax
Maximum cached transfer	MAXIMUM_CACHED_TRANSFER=32* MAXIMUM_CACHED_TRANSFER=*n*
Read cache	READ_CACHE* NOREAD_CACHE
Write-back cache	NOWRITEBACK_CACHE* WRITEBACK_CACHE
Availability	RUN* NORUN
Write protection	NOWRITE_PROTECT* WRITE_PROTECT

6. Verify the mirrorset configuration and switches:

   ```
   CLI> SHOW mirrorset-name
   ```

7. Verify the unit configuration and switches:

   ```
   CLI> SHOW unit-number
   ```

For example

The following example shows the commands you would use to create Mirr1, a two-member stripeset.

```
CLI> ADD MIRRORSET Mirr1 disk100 disk200
CLI> INITIALIZE Mirr1
CLI> ADD UNIT D200 Mirr1
CLI> SET D200 WRITEBACK_CACHE
CLI> SHOW Mirr1
CLI> SHOW D200
```

Configuring a RAIDset

See Chapter 3 for information about creating a profile and understanding the switches you can set for this kind of storage unit.

1. Create the RAIDset by adding its name to the controller's list of storagesets and specifying the disk drives it contains:

   CLI> **ADD RAIDSET** ***RAIDset-name DISKnnn DISKnnn DISKnnn***

2. Optional: set the RAIDset switches or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   CLI> **SET** ***RAIDset-name SWITCH_VALUE***

Table 5—6 RAIDset switches

RAIDset switch	Value and syntax
Replacement policy	POLICY=BEST_PERFORMANCE* POLICY=BEST_FIT NOPOLICY
Reconstruction speed	RECONSTRUCT=NORMAL* RECONSTRUCT=FAST

3. Initialize the RAIDset. Optional: if you want to set the Initialize switches, you must do so in this step:

   CLI> **INITIALIZE** ***RAIDset-name SWITCH_VALUE***

   Use the values in this table:

Table 5—7 Initialize switches

Initialize switch	Value and syntax
Chunk size	CHUNKSIZE=dEFAULT* CHUNKSIZE=*n*
Save configuration	NOSAVE_CONFIGURATION* SAVE_CONFIGURATION

4. Present the RAIDset to the host by giving it a unit number the host can recognize:

   CLI> **ADD UNIT** ***unit-number RAIDset-name***

5. Optional: set the Unit switches or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   ```
   CLI> SET unit-number SWITCH_VALUE
   ```

Table 5—8 Unit switches

Unit switch	Value and syntax
Maximum cached transfer	MAXIMUM_CACHED_TRANSFER=32* MAXIMUM_CACHED_TRANSFER=*n*
Read cache	READ_CACHE* NOREAD_CACHE
Write-back cache	NOWRITEBACK_CACHE* WRITEBACK_CACHE
Availability	RUN* NORUN
Write protection	NOWRITE_PROTECT* WRITE_PROTECT

6. Verify the RAIDset configuration and switches:

   ```
   CLI> SHOW RAIDset-name
   ```

7. Verify the unit configuration and switches:

   ```
   CLI> SHOW unit-number
   ```

For example

The following example shows the commands you would use to create Raid1, a three-member RAIDset.

```
CLI> ADD RAIDSET Raid1 disk100 disk200 disk300
CLI> INITIALIZE Raid1
CLI> ADD UNIT D300 Raid1
CLI> SET D300 WRITEBACK_CACHE
CLI> SHOW Raid1
CLI> SHOW D300
```

Configuring a striped mirrorset

See Chapter 3 for information about creating a profile and understanding the switches you can set for this kind of storage unit.

1. Create—but don't initialize—at least two mirrorsets.
2. Create a stripeset and specify the mirrorsets it contains:

   CLI> **ADD STRIPESET** ***mirrorset_1 mirrorset_2***

3. Initialize the stripeset. If you want to set any Initialize switches, you must do so in this step:

   CLI> **INITIALIZE** ***stripeset-name SWITCH_VALUE***

Table 5—9 Initialize switches

Initialize switch	Value and syntax
Chunk size	CHUNKSIZE=DEFAULT* CHUNKSIZE=*n*
Save configuration	NOSAVE_CONFIGURATION* SAVE_CONFIGURATION
Destroy	NODESTROY* DESTROY

4. Present the stripeset to the host by giving it a unit number the host can recognize:

   CLI> **ADD UNIT** ***unit-number stripeset-name***

5. Optional: set the Unit switches or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   CLI> **SET** ***unit-number SWITCH_VALUE***

Table 5—10 Unit switches

Unit switch	Value and syntax
Maximum cached transfer	MAXIMUM_CACHED_TRANSFER=32* MAXIMUM_CACHED_TRANSFER=*n*
Read cache	READ_CACHE* NOREAD_CACHE
Write-back cache	NOWRITEBACK_CACHE* WRITEBACK_CACHE
Availability	RUN* NORUN
Write protection	NOWRITE_PROTECT* WRITE_PROTECT

6. Verify the striped mirrorset configuration and switches:

   CLI> **SHOW *stripeset-name***

7. Verify the unit configuration and switches:

   CLI> **SHOW *unit-number***

For example

The following example shows the commands you would use to create Stripe1, a three-member striped mirrorset that comprises Mirr1, Mirr2, and Mirr3, each of which is a two-member mirrorset.

```
CLI> ADD MIRRORSET Mirr1 disk100 disk200
CLI> ADD MIRRORSET Mirr2 disk300 disk400
CLI> ADD MIRRORSET Mirr3 disk500 disk600
CLI> ADD STRIPESET Stripe1 Mirr1 Mirr2 Mirr3
CLI> INITIALIZE Stripe1 CHUNKSIZE=default
CLI> ADD UNIT D101 Stripe1
CLI> SET D101 WRITEBACK_CACHE
CLI> SHOW Stripe1
CLI> SHOW D101
```

Configuring a single-disk unit

Follow these steps to use a single disk drive as a single -disk unit in your subsystem.

1. Add the disk drive by following the steps for *Adding disk drives* on page 5–2.
2. Optional: set the device switches for the disk drive or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   CLI> **SET *DISKnnn SWITCH_VALUE***

Table 5—11 Device switches

Device switch	Value and syntax
Transportability	NOTRANSPORTABLE* TRANSPORTABLE
Transfer rate	TRANSFER_RATE_REQUESTED=10MHZ* TRANSFER_RATE_REQUESTED=5MHZ TRANSFER_RATE_REQUESTED=ASYNCHRONOUS

Note

Refer to Table 2-1 to determine the correct transfer rate for the device you are adding.

3. Present the disk drive to the host by giving it a unit number the host can recognize:

   CLI> **ADD UNIT *unit-number DISKnnn***

4. Optional: set the Unit switches or skip this step to accept the defaults(*). For each switch you want to set, enter the following command:

   CLI> **SET *unit-number SWITCH_VALUE***

Table 5—12 Unit switches

Unit switch	Value and syntax
Maximum cached transfer	MAXIMUM_CACHED_TRANSFER=32* MAXIMUM_CACHED_TRANSFER=*n*
Read cache	READ_CACHE* NOREAD_CACHE
Write-back cache	NOWRITEBACK_CACHE* WRITEBACK_CACHE
Availability	RUN* NORUN
Write protection	NOWRITE_PROTECT* WRITE_PROTECT

6. Verify the configuration:

```
CLI> SHOW DEVICES
```

For example

The following example shows the commands you would use to configure DISK100 as a single-disk unit.

```
CLI> ADD DISK DISK100 1 0 0
CLI> ADD UNIT D101 disk100
CLI> SET D101 SAVE_CONFIGURATION
CLI> SHOW DEVICES
```

Configuring a tape drive

The controller uses a passthrough device to transport the host's commands to and from a SCSI device such as a tape drive. For this reason, configuring a tape drive involves creating a passthrough device, then associating that device with the tape drive.

1. Create a passthrough device to logically represent the tape drive:

   `CLI>` **`ADD PASSTHROUGH`** ***`passthrough-name PTL-location`***

2. Present the passthrough device to the host by giving it a unit number the host can recognize:

   `CLI>` **`ADD UNIT`** ***`unit-number passthrough-name`***

3. Verify the configuration:

   `CLI>` **`SHOW DEVICES`**

For example

The following example shows the commands you would use to create a passthrough device for controlling a tape drive.

```
CLI> ADD PASSTHROUGH Pass100 1 0 0
CLI> ADD UNIT P100 Pass100
CLI> SHOW DEVICES
```

Configuring a tape loader

The controller uses a passthrough device to transport the host's commands to and from a SCSI device such as a tape loader. For this reason, configuring a tape loader involves creating a passthrough device, then associating that device with the loader.

1. Install and configure the tape drive.
2. Install the tape loader.
3. Create a passthrough device at the loader's PTL location to logically represent the tape loader:

   `CLI>` **`ADD PASSTHROUGH`** ***`passthrough-name PTL-location`***

4. Present the passthrough device to the host by giving it a unit number the host can recognize:

   `CLI>` **`ADD UNIT`** ***`unit-number loader-name`***

5. Verify the configuration:

   CLI> **SHOW DEVICES**

6. Install and configure the host-based software that controls the loader. (This software is not provided with your HS array controller or its software.)

For example

The following example shows the commands you would use to create a passthrough device for controlling a tape loader.

```
CLI> ADD PASSTHROUGH Pass130 1 3 0
CLI> ADD UNIT P130 Pass130
CLI> SHOW PASSTHROUGH
```

Partitioning a storageset or disk drive

See *Planning your partitions* in Chapter 3 for information about partitioning a storage unit.

1. Add the storageset or disk drive to the controller's list of storagesets and specify the disk drives it contains:

   CLI> **ADD *STORAGESET storageset-name DISKnnn DISKnnn***

   or

   CLI> **ADD DISK *DISKnnn PTL-location***

2. Initialize the storageset or disk drive. If you want to set any Initialize switches, you must do so in this step:

   CLI> **INITIALIZE *storageset-name***

3. Create each partition in the storageset or disk drive by indicating the partition's size:

   CLI> **CREATE_PARTITION *storageset-name* SIZE=*n***

   where *n* is the percentage of the disk drive or storageset that will be assigned to the partition. Enter SIZE=LARGEST to let the controller assign the largest free space available to the partition.

4. Verify the partitions:

   CLI> **SHOW *storageset-name***

The partition number appears in the first column, followed by the size and starting block of each partition.

5. Present each partition to the host by giving it a unit number the host can recognize. You can skip this step until you're ready to put the partitions online:

```
CLI> ADD UNIT unit-number storageset-name
     PARTITION=partition-number
```

6. Verify the unit numbers for the partitions:

```
CLI> SHOW storageset-name
```

7. Optional: set the Unit switches for each partition, or skip this step to accept the defaults(*). For each switch you want to set:

```
CLI> SET unit-number VALUE
```

Table 5—13 Syntax for Unit switches

Unit switch	Value and syntax
Maximum cached transfer	MAXIMUM_CACHED_TRANSFER=32* MAXIMUM_CACHED_TRANSFER=*n*
Read cache	READ_CACHE* NOREAD_CACHE
Write-back cache	NOWRITEBACK_CACHE* WRITEBACK_CACHE
Availability	RUN* NORUN
Write protection	NOWRITE_PROTECT* WRITE_PROTECT

For example

The following example shows the commands you would use to create Raid1, a three-member RAIDset, then partition into four storage units.

```
CLI> ADD RAIDSET Raid1 disk100 disk200 disk300
CLI> INITIALIZE Raid1
CLI> CREATE_PARTITION Raid1 SIZE=25
CLI> CREATE_PARTITION Raid1 SIZE=25
CLI> CREATE_PARTITION Raid1 SIZE=25
CLI> CREATE_PARTITION Raid1 SIZE=LARGEST
CLI> SHOW Raid1
.
.
.
Partition number   Size           Starting Block   Used by
----------------------------------------------------------
1                  1915 (0.98 MB) 0
2                  1915 (0.98 MB) 1920
3                  1915 (0.98 MB) 3840
4                  2371 (1.21 MB) 5760
.
.
.
CLI> ADD UNIT D1 Raid1 PARTITION=1
CLI> ADD UNIT D2 Raid1 PARTITION=2
CLI> ADD UNIT D3 Raid1 PARTITION=3
CLI> ADD UNIT D4 Raid1 PARTITION=4
CLI> SHOW Raid1
.
.
.
Partition number   Size           Starting Block   Used by
----------------------------------------------------------
1                  1915 (0.98 MB) 0                D1
2                  1915 (0.98 MB) 1920             D2
3                  1915 (0.98 MB) 3840             D3
4                  2371 (1.21 MB) 5760             D4
.
.
.
CLI> SET D1 WRITEBACK_CACHE
CLI> SET D2 WRITEBACK_CACHE
CLI> SET D3 WRITEBACK_CACHE
CLI> SET D4 WRITEBACK_CACHE
```

Adding a disk drive to the spareset

The spareset is a collection of hot spares that are available to the controller should it need to replace a failed member of a RAIDset or mirrorset.

Follow these steps to add a disk drive to the spareset:

1. Add the disk drive to the controller's spareset list:

   CLI> **ADD SPARESET** ***DISKnnn***

2. Repeat step 1 for each disk drive you want to add to the spareset:

3. Verify the contents of the spareset:

   CLI> **SHOW SPARESET**

For example

The following example shows the commands you would use to add DISK600 and DISK610 to the spareset.

```
CLI> ADD SPARESET disk600
CLI> ADD SPARESET disk610
CLI> SHOW SPARESET
```

Removing a disk drive from the spareset

You can't delete the spareset—it always exists whether or not it contains disk drives. However, you can delete disks in the spareset if you need to use them elsewhere in your StorageWorks subsystem.

1. Show the contents of the spareset:

   ```
   CLI> SHOW SPARESET
   ```

2. Delete the desired disk drive:

   ```
   CLI> DELETE SPARESET DISKnnn
   ```

3. Verify the contents of the spareset:

   ```
   CLI> SHOW SPARESET
   ```

For example

The following example shows the commands you would use to remove DISK600 from the spareset.

```
CLI> SHOW SPARESET

Name       Storageset     Uses             Used by
--------------------------------------------------
SPARESET   spareset       disk600
                          disk610


CLI> DELETE SPARESET disk600
CLI> SHOW SPARESET

Name       Storageset      Uses            Used by
--------------------------------------------------
SPARESET   spareset        disk610
```

Enabling Autospare

With AUTOSPARE enabled, any new disk drive that's inserted into the PTL location of a failed disk drive is automatically initialized and placed into the spareset. If initialization fails, the disk drive is put into the failedset .

To enable autospare:

CLI> **SET FAILEDSET AUTOSPARE**

To disable autospare:

CLI> **SET FAILEDSET NOAUTOSPARE**

During initialization, AUTOSPARE checks to see if the new disk drive contains metadata—the information that indicates it belongs to, or has been used by, a known storageset. If the disk drive contains metadata, initialization stops. (A new disk drive won't contain metadata but a repaired disk drive might. To erase metadata from a disk drive, add it to the controller's list of devices, then SET it to be TRANSPORTABLE.)

Deleting a storageset

Follow these steps to delete a storageset:

1. Show the configuration:

   CLI> **SHOW STORAGESETS**

2. Delete the unit number shown in the "Used by" column:

   CLI> **DELETE *unit-number***

3. Delete the name shown in the "Name" column:

   CLI> **DELETE *storageset-name***

4. Verify the configuration:

   CLI> **SHOW STORAGESETS**

For example

The following example shows the commands you would use to delete Stripe1, a three-member stripeset that comprises DISK100, DISK200, and DISK300.

```
CLI> SHOW STORAGESETS

Name       Storageset     Uses       Used by
---------------------------------------------
STRIPE1    stripeset      DISK100    D100
                          DISK200
                          DISK300

CLI> DELETE D100
CLI> DELETE Stripe1
CLI> SHOW STORAGESETS
```

Changing switches for a storageset or device

You can optimize a storageset or device at any time by changing the switches that are associated with it. See *Choosing switches for your storagesets and devices* in Chapter 3 for an explanation of the switches.

Note

Remember to update the storageset's profile when you change its switches.

Displaying the current switches

To display the current switches for a storageset or single-disk unit, enter the following command at a CLI prompt:

CLI> **SHOW *storageset-name* FULL**

Changing RAIDset and Mirrorset switches

Use the SET *storageset-name* command to change the RAIDset and Mirrorset switches associated with an existing storageset. For example, the following command changes the replacement policy for RAIDset Raid1 to best fit:

CLI> **SET Raid1 POLICY=best_fit**

Changing Device switches

Use the SET command to change the device switches. For example, the following command enables DISK100 to be used in a non-StorageWorks environment:

```
CLI> SET DISK100 TRANSPORTABLE
```

Changing Initialize switches

The Initialize switches can’t be changed without destroying the data on the storageset or device. These switches are integral to the formatting and can only be changed by re-initializing the storageset. Initializing a storageset is similar to formatting a disk drive; all data is destroyed during this procedure.

Changing Unit switches

Use the SET command to change Unit switches that are associated with a storageset. For example, the following command enables write protection for unit D100:

```
CLI> SET D100 WRITE_PROTECT
```

B

Working in OpenVMS systems

You'll want to be aware of the following issues if you're connecting your subsystem to an OpenVMS host.

Establishing a remote connection OpenVMS

After setting the controller's initial configuration, use HSZterm or a VAXcluster Console System (VCS) to communicate with the controller remotely from a VMS™ host.

Starting an HSZterm session

To create an HSZterm session, enter the following command at the DCL prompt:

$ **SET HOST/SCSI** ***device_name***

where device_name is the DK name of one of the devices or storagesets on the controller you want to connect to (for example, 1DKA401). A copyright notice and CLI prompt appear, indicating that you've established a remote connection to the controller.

Use the /LOG qualifier to create a log file of your sessions:

$ **SET HOST/SCSI/LOG=LOG.INFO*****device_name***

Starting a VCS terminal session

To communicate with the controller through a VCS terminal session, follow the instructions provided in the *VAXcluster Console System User's Guide.*

OpenVMS disk capacity limitations

OpenVMS VAX Versions 5.5-2 and earlier don't support disk capacities larger than 16,772,216 blocks (about 8.5 GB) as file-structured devices. You must keep this in mind when creating storage units, since stripesets and RAIDsets can easily exceed this limit.

The HSOF Version 5.0 software enforces a maximum byte count for ERASE commands of 4,194,303 blocks (about 2 GB). OpenVMS facilities that rely on ERASE commands automatically adjust to this behavior. It is only a concern for applications that issue ERASE commands directly.

Using storagesets as quorum disks

You can use any storage unit as a quorum disk in a SCSI or VAXcluster system. These include RAIDsets, mirrorsets, stripesets, striped mirrorsets, and single disk units.

Creating host-based shadow sets

Note

The save_configuration switch affects the capacities of the disk drives in a storageset. Thus, all or none of the storagesets you want to combine into a host-based shadow set must be initialized with the save_configuration switch.

Host-based shadow sets may only use storage units that comprise the same device types and device capacities. For example, you can create a host-based shadow set from two stripesets of RZ26 disk drives, but you can't create a shadow set from a stripeset of RZ26 disk drives and a stripeset of RZ74 disk drives.

C

Working in DIGITAL UNIX systems

You'll want to be aware of the following issues if you're connecting your subsystem to a DIGITAL UNIX host.

Establishing a remote connection on DIGITAL UNIX

After setting the controller's initial configuration, use HSZterm to communicate with the controller remotely from a DIGITAL UNIX host.

HSZterm supports all of the CLI commands and most of the local programs. It doesn't support local programs that use cursor-positioning escape sequences, such as VTDPY.

HSZterm runs on all systems that support the DIGITAL UNIX Operating System Version 2.0 or Version 3.0. See the *System Manager's Guide for HSZterm* for instructions on using HSZterm on a DIGITAL UNIX host.

Creating device special files

You'll need to create block and character special files b efore a storageset, single-disk unit, or other storage device can be accessed by the host. All eight host partitions of a disk-based storage unit must have special files located in the /dev directory.

Follow these general steps to create the device special files for a disk-based storage unit. They're described in detail on the next few pages:

1. Create a DIGITAL UNIX block special device name based on the storageset's unit number and the host's SCSI bus number to which the controller is attached.
2. Generate the block and character special files with the MAKE_RAID_LUNS utility. If you don't have DIGITAL UNIX Version 3.2A or later, you'll have to use the mknod utility instead.

Creating DIGITAL UNIX block special device names

DIGITAL UNIX does not enforce a device-naming format, however, DIGITAL recommends you use *rzxnny* for block special files and *rrzxnny* for character special files,[1] where:

- *rz* denotes a block special device name and *rrz* denotes a character special device name.
- *x* is the letter "a" through "h" that corresponds to the last digit in the storageset's unit number—the storageset's ID or LUN. Use a, b, c, d... for 0, 1, 2, 3... respectively.
- *nn* is the device number, which is derived as
  (8 * host's SCSI bus #) + (controller's target ID)

  The controller's target ID is the first digit of the unit number. If it's a single-digit unit number, such as D1, the controller's target ID is zero.
- *y* is the letter "a" through "h" that indicates the device partition as seen by the host. You only need to specify a partition letter if you're using the mknod utility to generate the block and character special files. The MAKE_RAID_LUNS utility creates these letters for you.

For example

The following example derives the block special device name for a storageset, unit D301, that's serviced by a controller connected to the host's SCSI bus 2. The storageset's block special device name is rzb19, which is derived:

rz + LUN letter + ((8*SCSI bus #) + controller's target ID)

or

rz + b + ((8 * 2) + 3)

or

rzb19c

---

[1] Some DIGITAL UNIX utilities, such as makedev, iostat, and certain startup procedures, will not recognize this format.

Using the MAKE_RAID_LUNS utility

Use the MAKE_RAID_LUNS utility to create the block and character special files for all eight host partitions of a storage unit. This utility is available in DIGITAL UNIX Versions 3.2A and later.

Each special file references the major and minor number of a specific partition. Once you've created the block and character special files for the storageset, you can use the DIGITAL UNIX device name to access the storageset through normal I/O system routines.

To create block and character special files for all host partitions for a storageset:

1. Change to the /dev directory:

   ```
   # cd /dev
   ```

2. Run the make_raid_luns utility on the block special device name you created for the storageset. Omit the partition letter; this utility creates it for you:

   ```
   # ./MAKE_RAID_LUNS block_special_device_name
   ```

For example

The following example creates block special files rzc19a through rzc19h and character special files rrzc19a through rrzc19a.

```
# cd /dev
# ./MAKE_RAID_LUNS rzc19
```

Using the mknod utility

Caution

> The mknod utility does not verify the minor number, and it does not signal any errors. If you use the wrong minor number, trying to access the device using the DIGITAL UNIX device name would yield unpredictable results, such as accessing the wrong partition, accessing the wrong controller unit, and so forth.

If your system doesn't have the MAKE_RAID_LUNS utility, you can create the special files using the mknod utility in the /usr/sbin directory.

To create the special files with the mknod utility, enter the following command at the system prompt:

/usr/sbin/mknod OSF/1device-name type major-number minor-number

where:

- *type* is "b" for block mode files or "c" for character mode files.
- *major number* for a disk-based storage unit is always 8.
- *minor number* is derived as: (16384 * host's SCSI bus number) + (1024 * controller's target ID) + (64 * storageset's ID or LUN) + the UNIX partition number.

For example

The following example creates block and character special files for storageset D200 that's accessed through a controller on the host's SCSI bus 2. Its block special device name is rza18a:

1. Calculate the minor number: (16384 * 2) + (1024 * 2) + (64 * 0) + 0 = 34816
2. Run the mknod utility to produce the block and character special files for the first host partition, rza18a and rrza18a:

```
# cd /dev
# /usr/sbin/mknod /dev/rza18a b 8 34816
# /usr/sbin/mknod /dev/rrza18a c 8 34816
```

3. Repeat steps 1 and 2 to create the special files for the remaining seven host partitions rza18(b-h) and rrza18(b-h).

Creating configuration file entries

You can access storagesets using the standard CAM driver without making entries in the configuration file. However, to see the storagesets from startup—and to make the output of the iostat utility easier to read —you'll need to create entries for all of the storagesets.

Use **genvmunix** to initialize the system and **doconfig** to build a new configuration file. The new configuration file will only list the LUN 0 storagesets.

Before rebuilding a configuration file, save any customized configuration file that has entries for storagesets. After rebuilding the configuration file, add these entries to the new configuration file. Alternatively, you could extract the storage unit entries and save them in a separate file. Then, after building the new configuration file with genvmunix, you could merge the saved information into the new configuration file.

Note

The configuration file name format is not the same as the format for device special files. You must use the *rznn* format in your configuration file, and it is recommended you use the *rzxnny* format for device special files. The different names do not conflict because they are used by different pieces of the operating system.

Entries for storagesets in the configuration file have the following format: “device disk*name* at scsi*z* drive*number*,” where:

- *name* is in the format rz*nn*, where *nn* is derived as: (8 * host’s SCSI bus #) + (controller’s target ID).
- *z* in the entry "at scsi*z*" is the host SCSI bus number
- *number* is a unique drive number for each controller unit and is calculated as: (64 * host SCSI #) + (8 * controller’s target ID) + storageset’s ID or LUN. You only need to calculate the drive number of the first device using the formula. You can then increment each subsequent LUN by 1.

For example

Note

Even if you don’t configure all eight LUNs on each controller target ID, you must put all eight entries in the configuration file. This makes the reports from certain utilities, such as iostat, more legible.

The following example shows the configuration file for storagesets D100 through D107. These storagesets are serviced by a controller on the host's SCSI bus 2.

```
bus    tcds0  at tc0 slot 6 vector       tcdsintr
bus    tza0   at tc0 slot 5 vector       tzaintr
controller    scsi2  at tza0      slot 0
device disk   rz16   at scsi2     drive  136
device disk   rz16   at scsi2     drive  137
device disk   rz16   at scsi2     drive  138
device disk   rz16   at scsi2     drive  139
device disk   rz16   at scsi2     drive  140
device disk   rz16   at scsi2     drive  141
device disk   rz16   at scsi2     drive  142
device disk   rz16   at scsi2     drive  143
```

Using storagesets as initialization devices

Any storageset whose unit number ends in zero can be used as a system initialization device. (If the unit number ends in zero its ID or LUN is zero.) After you've configured the storageset using CFMENU or the CLI, install the DIGITAL UNIX operating system on the unit.

DECsafe Available Server Environment

You can use disk devices with the DECsafe Available Server Environment (ASE) for DIGITAL UNIX, provided you use a valid host configuration (including host adapters) to support them. Refer to the *DIGITAL UNIX ASE Installation and User's Guide* for further information. See the release notes for supported host adapters and DIGITAL UNIX version levels for ASE.

Using DIGITAL UNIX utilities

This section provides notes on the interaction of the following DIGITAL UNIX utilities with storagesets in your subsystem: file, disklabel, SCU, iostat, and uerf.

file

You can use the DIGITAL UNIX file utility to determine if a storage unit can be accessed from the host. If the test is successful, the green LED on the device will flash and some information about the unit will be displayed on the console.

The unit you want to test must already have a character special file and the correct disk label.

For example, to check the accessibility of unit number D101:

1. Disable the read cache for the unit. If read caching is not disabled, the data required by the file command may be in the cache and the unit will not be accessed.

   ```
   CLI> SET D101 NOREAD
   ```

2. Run the file command and specify the character mode device special file, such as:

   ```
   # /usr/bin/file /dev/rrzb17a
   ```

   The device's green LED will illuminate. If the storage unit contains more than one disk drive, the LED flashes on and then off very quickly on only one of the disk drives. The DIGITAL UNIX operating system should display something like the following output after the command is entered:

   ```
   /dev/rrzb17a character special (8/33856) SCSI #2
   «Controller» disk #146 (SCSI ID #1)
   ```

   where: 8 is the major number; 33856 is the minor number; 2 is the host's SCSI bus number; 146 is the drive number as listed in the Configuration File; and 1 is the controller's target ID.

   If you get the following message, it usually means that the special file that matches the minor number does not exist in the /dev directory:

   ```
   file: Cannot get file status on /dev/33856
   /dev/33856: cannot open for reading
   ```

   If the only output that is returned from the file command is the major and minor number, then either the device is not answering or the device special file does not have the correct minor number. Check the minor number to be sure that it matches the host SCSI bus number, the controller target ID, and the LUN of controller unit.

If an error occurs regarding the disk label, there is a good probability that the device can be accessed. This error can usually be fixed by creating the disk label with the DIGITAL UNIX disklabel utility.

3. Re-enable the unit's read cache when you’re done testing the storageset’s accessibility:

   ```
   «Controller»> SET D101 READ
   ```

disklabel

The disklabel utility looks in the /mdec directory for certain files for each device. If you receive bootblock errors when using disklabel, you must create these files with the following commands:

```
# ln /mdec/bootrz /mdec/bootrza
# ln /mdec/bootrz /mdec/bootrzb
# ln /mdec/bootrz /mdec/bootrzc
# ln /mdec/bootrz /mdec/bootrzd
# ln /mdec/bootrz /mdec/bootrze
# ln /mdec/bootrz /mdec/bootrzf
# ln /mdec/bootrz /mdec/bootrzg
# ln /mdec/bootrz /mdec/bootrzh

# ln /mdec/rzboot /mdec/rzaboot
# ln /mdec/rzboot /mdec/rzbboot
# ln /mdec/rzboot /mdec/rzcboot
# ln /mdec/rzboot /mdec/rzdboot
# ln /mdec/rzboot /mdec/rzeboot
# ln /mdec/rzboot /mdec/rzfboot
# ln /mdec/rzboot /mdec/rzgboot
# ln /mdec/rzboot /mdec/rzhboot
```

SCU

You can use the SCSI CAM Utility (SCU) to see which storagesets are available to the DIGITAL UNIX operating system:

```
# /sbin/scu -f /dev/character_special_file
```

Use the show nexus SCU command to get information about device locations:

```
SCU> show nexus

     Device Nexus: Bus: n
     Target: t
     Lun: L
     Device Type - direct access
```

Use the scan edt SCU command to poll all devices on the host-side SCSI buses. This allows you to show what devices are available from all host-side SCSI buses. The device special files do not have to exist for SCU to see the devices:

```
SCU> scan edt
SCU> show edt

 CAM Equipment Device Table (EDT) Information:
 Bus: 2, Target: 1, Lun: 0, Device Type: Direct  Access
 Bus: 2, Target: 1, Lun: 1, Device Type: Direct Access
 Bus: 2, Target: 1, Lun: 2, Device Type: Direct Access
 Bus: 2, Target: 1, Lun: 3, Device Type: Direct Access
 Bus: 2, Target: 1, Lun: 4, Device Type: Direct Access
 Bus: 2, Target: 1, Lun: 5, Device Type: Direct Access
 Bus: 2, Target: 1, Lun: 6, Device Type: Direct Access
 Bus: 2, Target: 1, Lun: 7, Device Type: Direct Access
```

Storagesets look like any other SCSI device. All eight entries for bus 2, target 1 are storagesets. For example, the last entry is for unit D107 on the host’s SCSI bus 2.

iostat

You can use the iostat utility to view performance statistics for storagesets. (Set your display to 132 columns before running iostat.)

To run iostat, enter the following command at the system prompt:

```
iostat rznn s t
```

where:

- *nn* is derived as (8 * host's SCSI bus #) + (controller's target ID).
- *s* is optional and denotes the amount of time, in seconds, between screen updates.
- *t* is optional and denotes the total number of screen updates.

The output from iostat shows all devices that have device name *rznn*. The information for LUN 0 is in the first column, LUN 1 in the second column, and so forth. It's much easier to interpret the output if the configuration file contains entries for all eight devices. If the configuration file doesn't contain entries for all devices, the iostat output has fewer columns and it is difficult to correlate each column with a specific device.

For example

The following example shows the activity for rzh16 on LUN7:

```
# iostat rz16 5 4

  rz16    rz16    rz16    rz16    rz16    rz16
bps tps bps tps bps tps bps tps bps tps bps tps
 0   0   0   0   0   0   0   0   0   0  126   3
 0   0   0   0   0   0   0   0   0   0 1618  34
 0   0   0   0   0   0   0   0   0   0 1639  34
 0   0   0   0   0   0   0   0   0   0 1610  34
```

uerf

The operating system logs events to the binary.errlog file, which you can access with the UNIX error report formatter (uerf).

Use uerf to show the controller model name and all of the extended sense data. Use the -Z switch to help display unsupported error entries. The data is displayed in hex format. The command format is:

```
# uerf -Z -o full -r 199 -R
```

The names of the routines and the nature of the problem are displayed in the ASCII representation portion of the hex data. The reporting component is the DEC SIMPORT and the DEC TZA SPO. This information points to the CAM component that detected the error and can be useful in isolating the problem.

D

Working in Windows NT systems

You'll want to be aware of the following issues if you're connecting your subsystem to a Windows NT system.

Establishing a remote connection on Windows NT

After setting the controller's initial configuration , use a terminal-emulation program such as Windows NT Terminal to communicate with a controller remotely—that is, through your host rather than through a terminal connected to the local-connection port on the front of the controller.

To establish a remote connection from a Windows NT host:

1. Connect a serial cable from the controller's local -connection port to a serial communication port on your host.
2. Start your terminal emulation program.
3. Configure the host's serial port for 9600 baud, 8 data bits, 1 stop bit, and no parity.
4. Press the Enter key. A copyright notice and CLI prompt appear, indicating that you've established a remote connection to the controller.

Accessing storage units from your host

After you've configured your controllers as described in Chapter 2, use the RAID Manager software included in your platform kit to configure the storage units in your subsystem. You can also use CFMENU or CLI to configure storage units in your subsystem.

Before your host can see the storage units in your subsystem, you'll have to reboot your system, then use the Windows Disk Administrator ® to partition and format each storage unit. After you've partitioned and formatted each storage unit, Windows NT sees them as single, large-capacity, disk drives. See your Windows NT documentation for instructions on using the Disk Administrator.

Windows NT assigns disk names based on the order in which the system drivers "find" the "disks" during initialization. In addition, the disk-class driver HSZDISK.SYS connects to all storagesets before it connects to any other disks in the system. Therefore, the first entries in the Disk Administrator should represent all of your storagesets.

The default names are Disk 0, Disk 1, and so on. The order of storagesets corresponds to their bus and unit numbers.

Verify that there is an entry in the Disk Administrator display for each of your storagesets and that their capacities are correct.

Changing or deleting storagesets

Before you change or delete a storage unit, you’ll need to delete its Windows NT disk partitions using Disk Administrator.

Glossary

adapter

A device that converts the protocol and hardware interface of one bus type into that of another without changing the functionality of the bus.

allocation class

A numerical value assigned to a controller to identify units across multiple, independent controllers. (Controllers in a dual-redundant configuration must have the same allocation class.)

array controller

A hardware/software device that facilitates communications between a host and one or more devices organized in an array. HS family controllers are examples of array controllers.

BBR

Bad block replacement. The procedure used to locate a replacement block, mark the bad block as replaced, and move the data from the bad block to the replacement block.

BBU

Battery backup unit. A StorageWorks SBB option that extends power availability after the loss of primary ac power or a power supply to protect against the corruption or loss of data.

block

The smallest data unit addressable on a disk. Also called a sector. In integrated storage elements, a block contains 512 bytes of data, EDC, ECC, flags, and the block's address header.

CDU

Cable distribution unit. The power entry device for StorageWorks cabinets. The unit provides the connections necessary to distribute ac power to cabinet shelves and fans.

CI bus

Computer interconnect bus. Uses two serial paths, each with a transfer rate of 70 Mb/s (8.75 MB/s).

CLI

Command line interpreter. Operator command line interface for the HS family controller firmware.

controller shelf

A StorageWorks shelf designed to contain controller and cache memory modules.

CRC

An 8-character cyclic redundancy check string used in conjunction with the customer identification string for turning on licensed features such as write-back caching.

data center cabinet

A generic reference to the large cabinets, such as the SW800-series, in which StorageWorks components can be mounted.

DDL

Dual data link. The ability to operate on the CI bus using both paths simultaneously to the same remote node.

differential SCSI bus

A signal’s level is determined by the potential difference between two wires. A differential bus is more robust and less subject to electrical noise than is a single-ended bus.

DILX

Disk inline exerciser. Diagnostic firmware used to test the data transfer capabilities of disk drives in a way that simulates a high level of user activity.

DSA

Digital storage architecture. A set of specifications and interfaces describing standards for designing mass storage products. DSA defines the functions performed by host computers, controllers, and disk drives. It also specifies how they interact to accomplish mass storage management.

DSSI

Digital storage system interconnect. A Digital-specific data bus with an 8-bit data transfer rate of 4 MB/s.

dual-Redundant configuration

Two controllers in one controller shelf providing the ability for one controller to take over the work of the other controller in the event of a failure of the other controller.

DUART

Dual universal asynchronous receiver/transmitter. An integrated circuit containing two serial, asynchronous transceiver circuits.

DUP

Diagnostic and utility protocol. Host application software that allows a host terminal to be connected to the controller's command line interpreter.

DWZZA

The StorageWorks compatible SCSI bus signal converter.

ECB

External cache battery.

ECC

Error correction code. One or more cyclic redundancy check (CRC) words that allow detection of a mismatch between transmitted and received data in a communications system, or between stored and retrieved data in a storage system. The ECC allows for location and correction of an error in the received/retrieved data. All ECCs have limited correction power.

EDC

Error detection code. One or more checksum words that allow detection of a mismatch between transmitted and received data in a communications system, or between stored and retrieved data in a storage system. The EDC has no data correction capability.

ESD

Electrostatic discharge. The discharge of a potentially harmful static electric voltage as a result of improper grounding.

failedset

A group of disk drives that have been removed from RAIDsets due to a failure or a manual removal. Disk drives in the failedset should be considered defective and should be tested, repaired, and then placed into the spareset.

failover

The process that takes place when one controller in a dual-redundant configuration assumes the workload of a failed controller.

flush

The act of writing data from the cache module to the media.

FRU

Field replaceable unit. A hardware component that can be replaced.

FWD SCSI

Fast, wide, differential SCSI. The differential SCSI bus with a 16-bit parallel data path that yields a transfer rate of up to 20 MB/s.

half-height device

A device that occupies half of a 5.25 inch SBB carrier. Two half-height devices can be mounted in a 5.25 inch SBB carrier. The first half-height device is normally mounted in the lower part of the carrier. The second device is normally mounted in the upper part of the carrier.

HBVS

Host-based volume shadowing. Also known as Phase 2 volume shadowing.

HSOF

Hierarchical storage operating firmware. Software contained on a program card that provides the logic for the HS array controllers.

HIS

Host interconnect services. The firmware in the HS array controller that communicates with the host.

host

Any computer to which a storage subsystem can be attached.

hot swap

A method of replacing a device whereby the system that contains the device remains online and active during replacement. The device being replaced is the only device that cannot perform operations during a hot swap.

initiator

A SCSI device that requests an I/O process to be performed by another SCSI device (a target). This is always the controller.

local terminal

A terminal plugged into the EIA-423 maintenance port on the front bezel of the HS array controller. Also called a maintenance terminal.

logical unit

The physical device or storage unit seen by the host. Often these logical units are spread across more than one physical device, especially in RAID implementations. This is *not* a LUN.

Logical Unit Number

See LUN.

LRU

Least recently used. This is cache terminology for the block replacement policy for the read cache.

LUN

A logical unit number is a physical or virtual peripheral device addressable through a target. LUNs use their target’s bus connection to communicate on the SCSI bus.

maintenance terminal

Any EIA-423 compatible terminal to be plugged into the HS controller. This terminal is used to identify the controller, enable host paths, define the configuration, and check controller status. It is not required for normal operations. It is sometimes referred to as a local terminal.

metadata

Data written on the physical disk that is not visible to the host/customer that allows the HS array controller to maintain a high integrity of customer data.

mirrorset

Two or more physical disks configured to present one highly reliable virtual unit to the host.

MSCP

Mass storage control protocol. The protocol by which blocks of information are transferred between the host and the controller.

non-redundant configuration

A single controller configuration. A controller configuration which does not include an second backup controller permitting failover in the event of a failure.

normal member

A mirrorset member whose entire contents is guaranteed to be the same as all other NORMAL members. All NORMAL members are exactly equivalent.

normalizing member

A mirrorset member whose contents is the same as all other NORMAL and NORMALIZING members for data that has been written since the mirrorset was created or lost cache data was cleared. Data that has never been written may differ among NORMALIZING members.

NV

Nonvolatile. A term used to describe memory that can retain data during a power loss to the controller.

partition

A percentage of a storageset or single-disk unit that may be presented to the host as a storage unit.

partitioning

Device partitioning is the process of dividing up a single large container into a number of smaller containers.

port

The hardware and software used to connect a host controller to a communication bus, such as CI, DSSI, or SCSI bus. This term also is used to describe the connection between the controller and its SCSI storage devices.

PTL

Port-target-LUN. A method of device notation where **P** designates the controller's device port (1-6), **T** designates the target ID of the device (0-6), and **L** designates the LUN of the device (0-7).

qualified device

A device that has been fully tested in an approved StorageWorks configuration, (that is, shelf, cabinet, power supply, cabling, and so forth) and is in complete compliance with country-specific standards (for example, FCC, TUV, and so forth) and with all Digital standards.

quiesce

To make a bus inactive or dormant. The operator must quiesce SCSI bus operations, for example, during a device warm swap.

RAID

Redundant array of independent disks. The multiple storage access methods devised for performance (RAID 0, striping) and/or various cost levels of availability (RAID 1 through RAID 5).

RAIDset

Three or more physical disks that are configured to present an array of disks as a single virtual unit to the host.

read cache

The cache used to accelerate read operations by retaining data which has been previously read, written, or erased, based on a prediction that it will be reread.

replacement policy

The method by which a spare disk is selected to replace a disk that has failed in a RAIDset.

SBB

StorageWorks building block. A modular carrier plus the individual mechanical and electromechanical interface required to mount it into a standard StorageWorks shelf. Any device conforming to shelf mechanical and electrical standards is considered an SBB.

SBB shelf

StorageWorks building block shelf. A StorageWorks shelf, such as the BA350--Sx, designed to house plug-in SBB modules.

SCS

System communication services. A delivery protocol for packets of information (commands or data) to or from the host.

SCSI

Small computer system interface. An ANSI interface defining the physical and electrical parameters of a parallel I/O bus used to connect initiators to a maximum of seven devices. The StorageWorks device interface is implemented according to SCSI--2 standard, allowing the synchronous transfer of 8-bit data at rates of up to 10 MB/s.

SCSI device

A host computer adapter, a peripheral controller, or a storage element that can be attached to the SCSI bus.

SCSI device ID

The bit-significant representation of the SCSI addressing that refers to one of the signal lines numbered 0 through 7. Also referred to as a *target* ID.

SCSI--A cable

A 50-conductor (25 twisted pair cable used for single-ended, SCSI bus connections.

SCSI--P cable

A 68-conductor (34 twisted pair cable used for differential bus connections.

Small Computer System Interface

See SCSI.

spareset

A pool of disk drives used by the controller to replace failed members of a RAIDset.

SPD

Software product description. A document that contains the legal description of a product.

storageset

Any collection of containers, such as stripesets, RAIDsets, the spareset, and the failedset, that make up a container.

storage unit

The general term that refers to storagesets, single disk units, and all other storage devices that can be installed in your subsystem and accessed by a host. A storage unit can be any entity that is capable of storing data, whether it is a physical device or a group of physical devices.

StorageWorks

Digital's family of modular data storage products that allows customers to design and configure their own storage subsystems. Components include power, packaging, cabling, devices, controllers, and software. Customers can integrate devices and array controllers in StorageWorks enclosure to form storage subsystems.

StorageWorks building block

See SBB.

stripeset

A virtual disk drive with its physical data spread across multiple physical disks. Stripeset configurations do not include a data recovery mechanism.

striped mirrorset

Stripesets whose members have been mirrored.

tagged command queuing

A SCSI feature that allows a device to have multiple I/O requests outstanding to it at one time.

target

A SCSI device that performs an operation requested by an initiator. The target number is determined by the device's address on its SCSI bus.

TILX

Tape inline exerciser. Diagnostic firmware used to test the data transfer capabilities of tape drives in a way that simulates a high level of user activity.

TMSCP

Tape mass storage control protocol. The protocol by which blocks of information are transferred between the host and the controller.

unit

The host's view of a container on an HS array controller. A unit may be made up of simply a physical disk or tape drive, or a more complex container such as a RAIDset.

unwritten cached data

Data in the write-back cache which has not yet been written to the physical device, but the user has been notified that the data has been written.

VCS

VAXcluster console system.

virtual terminal

A software path from an operator terminal on the host to the controller's CLI. The path can be established via the host port on the controller (using DUP) or via the maintenance port through on intermediary host (VCS). A virtual terminal is also sometimes called a host console.

warm swap

A method for adding or replacing a device whereby the system remains online, but all activity on the device's bus must be halted for the duration of the swap.

write-back caching

A caching strategy that writes data to the cache memory, then flushes the data to the intended device at some future time. From the user's perspective, the write operation is complete when the data is stored in the cache memory. This strategy avoids unnecessary access of the devices.

write hole

Undetectable RAID level 1 or 5 data corruption. A write hole is caused by the successful writing of some, but not all, of the storageset members. Write holes occur under conditions such as power outages, where the writing of multiple members can be abruptly interrupted. A battery backed-up cache design eliminates the write hole, because data is preserved and writes can be retried.

write-through cache

A cache write strategy in which the destination of the write data is the primary storage media. This operation may update, invalidate, or delete data from the cache memory accordingly, to ensure that the cache does not contain obsolete data. The user sees the operation as complete only after the backup storage device has been updated.

Index

A

B

C

D

E

F

G

H

I

T

U