VMS

digital

Guide to Maintaining a VMS System

Order Number AA–LA34A–TE

Guide to Maintaining a VMS System

Order Number: AA–LA34A–TE

April 1988

This manual provides system managers and operators with the concepts and procedures needed to maintain daily operations on a VMS operating system.

Revision/Update Information:	This is a new manual.
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

ZK3386

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

EXAMPLES

FIGURES

TABLES

Preface

The *Guide to Maintaining a VMS System* provides system managers with guidelines and procedures needed to maintain daily operations on a VMS operating system. After reading this manual, system managers should be able to perform the following tasks:

- Set up public volumes
- Handle user requests for disk and tape operations
- Backup files and volumes
- Manage disk space
- Manage batch and print operations
- Maintain system log files

This manual is not intended to be a complete one-volume source of information. For information on customizing the system after installation and setting up user accounts, see the *Guide to Setting Up a VMS System*. The utilities and commands used to perform specific system management tasks are described in detail in the individual VMS utility manuals and in the *VMS DCL Dictionary*.

Intended Audience

This manual addresses experienced users of the VMS operating system who perform the functions of a system manager or operator.

Document Structure

The *Guide to Maintaining a VMS System* contains the following task-oriented chapters, each describing a different system management function:

- **Chapter 1** - Introduction
- **Chapter 2** - Setting Up and Maintaining Public Volumes
- **Chapter 3** - Performing Disk and Magnetic Tape Operations
- **Chapter 4** - Backing Up Files and Volumes
- **Chapter 5** - Managing Disk Space
- **Chapter 6** - Performing Batch and Print Operations
- **Chapter 7** - Maintaining System Log Files

Associated Documents

- For general background information about the system, see the *Introduction to VMS.*
- For instructions for performing processor-specific procedures (for example, installation, bootstrap operations, and backing up the console medium), see your VAX processor installation and operations guide.
- For information on setting up your system for daily operations, see the *Guide to Setting Up a VMS System.*
- For information on creating and maintaining volumes using the volume shadowing option, see the *VAX Volume Shadowing Manual.*
- For hardware operating instructions, see the appropriate hardware owner's manual.
- For managing network operations, see the *Guide to DECnet–VAX Networking.*
- For configuring and managing VAXclusters, see the *VMS VAXcluster Manual.*
- For information on performance tuning, see the *Guide to VMS Performance Management.*
- For information about the new System Management (SYSMAN) Utility, see the *VMS SYSMAN Utility Manual.* For detailed information on other VMS utilities, see the specific VMS utility manual.
- For supplemental reference information, see the *VMS DCL Dictionary* and the *VMS System Messages and Recovery Procedures Reference Volume.*

Conventions

Convention	Meaning
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

New and Changed Features

- New System Management Utility (SYSMAN) available for facilitating the management of nodes and clusters.
- New and Enhanced Batch/Print features:
  - New DCL command SHOW ENTRY with the following qualifier options: /BATCH, /BRIEF, /DEVICE, /FILES, /FULL, /OUTPUT, /USER, /BY_JOB_STATUS, /GENERIC
  - New DCL command SET ENTRY replaces SET QUEUE/ENTRY
  - Enhancements to the SHOW QUEUE command, including new qualifiers: /SUMMARY, /BY_JOB_STATUS, /GENERIC
  - Changes in the functionality of the /ENTRY and /DEVICE qualifiers
  - New F$GETQUI lexical function
  - New DCL qualifiers: SET QUEUE (/OPEN, /CLOSE)
  - New job and queue status messages
  - New ACL-based queue protection, using the DCL command SET ACL/OBJECT_TYPE=JOBCTL_QUEUE
- The ADDUSER.COM, BACKUSER.COM and RESTUSER.COM command procedures, previously located in the SYS$MANAGER directory, are now available in the SYS$EXAMPLES directory. They can be used as templates for adding users to your system, backing up files, and restoring files.
- Mount Verification procedure enhanced to handle magnetic tapes
- Backup Utility enhancements

1 Introduction

This manual provides guidelines and procedures for maintaining daily operations on a VMS operating system. Maintenance tasks may be performed by an operator, a system manager, or several people. The following is an overview of the routine maintenance tasks described in this manual:

- **Setting up and maintaining public volumes**

  A primary system management duty is setting up and maintaining public volumes. Public volumes, also called system volumes, are file-structured disk volumes that contain public files. Such files are made available to most, if not all, users. Public volumes can also contain files that individuals create for their own use. As system manager, you should find the balance between your users' needs and the system's available mass storage resources. You must determine how mass storage devices on your system are to be configured, which devices will hold public system volumes, which devices will be available for user's private volumes, and how the public volumes will be configured. See Chapter 2 for detailed information.

- **Handling user requests for disk and magnetic tape operations**

  Users may prepare and manipulate their own volumes, or they may rely on assistance from an operator. If users require assistance, they can use the operator communication manager (OPCOM), an online facility for communication between operator and user. See Chapter 3 for more information on handling user requests with OPCOM.

- **Backing up public files and volumes**

  One of the best ways to reduce the risk of losing data on your system is to regularly back up your files and volumes. If you have limited disk space, it is practical to store seldom-used files on magnetic tape until you need them. You use the BACKUP Utility by entering specific qualifiers to the DCL command BACKUP. See Chapter 4 for more information about the BACKUP command.

- **Managing disk space**

  You control the amount of disk space available to users by creating and maintaining quota files on public volumes with the disk quota subcommand of the SYSMAN Utility. The absolute maximum number of blocks allotted to a user on a volume is the sum of the quota and overdraft values. In addition to maintaining disk quotas, you can also conserve disk space by periodically purging log files and setting file expiration dates. See Chapter 5 for more information.

- **Performing batch and print operations**

  Setting up queues and managing batch and print jobs is of primary importance to maintaining daily operations on your system. As system manager, you must monitor the types of jobs that run on your system and develop ways to submit, schedule, and execute jobs efficiently. After you have set up the queues, you must monitor performance and trouble shoot for potential log jams in the processing of jobs in the queues. For example, you may need to delete a job entry, or pause a queue temporarily to change the printer form or correct a defective printer device. You interact with the batch/print facility by entering DCL commands. Most of the commands for managing queues require privileges (see the *VMS DCL Dictionary* for information on privileges required for specific commands). See Chapter 6 for guidelines and procedures for performing batch and print operations.

- **Maintaining system log files**

  The VMS operating system provides several system log files to assist you in monitoring system performance and analyzing errors and other system events. These log files include the following:

  System dump file
  Error log file
  Operator log file
  Accounting log file

  In the event of a severe system failure, the VMS operating system automatically shuts down and produces a crash dump of the state of the system at the time that the error was detected. The system dump file assists you in analyzing the cause of the system failure. (See the *VMS System Dump Analyzer Utility Manual* for more information on the System Dump Analyzer; see Chapter 7 for more information on the system log files.)

If you are manager of a cluster, or of a system with multiple nodes, you should also become familiar with the System Management (SYSMAN) Utility, which centralizes the management of nodes and clusters. In a multinode environment, SYSMAN allows you to define your management environment to be a particular node, a group of nodes, or a cluster. After you define the management environment, you can perform system management tasks on all nodes in the target environment from your local node, without having to log in to each individual node.

Using SYSMAN to maintain your system is very similar to using the traditional system management tools. SYSMAN uses the software tools with which you may already be familiar, including DCL commands such as MOUNT, DEFINE, INITIALIZE, SET, and SHOW. SYSMAN also executes VMS system management utilities, such as AUTHORIZE, AUTOGEN, and INSTALL. In addition to giving you access to these system management tools, SYSMAN allows you to set disk quotas (using the DISKQUOTA command set, which operated as a standalone utility prior to VMS Version 5.0), to modify system parameters (with the same functions as SYSGEN), and to build site-specific startup procedures.

For more information about the SYSMAN Utility, see the *VMS SYSMAN Utility Manual.*

2 Setting Up and Maintaining Public Volumes

This chapter contains guidelines for setting up and maintaining public files and volumes. It also describes how to initialize and mount public volumes, and how to transfer information between VMS Files–11 formatted and *foreign* (non-Files–11 formatted) volumes. For information on mounting tapes and disks or backing up your files, see Chapter 3 or 4, respectively.

2.1 Planning Public Volumes

Public volumes are file-structured disk volumes that contain user and system files. Such files must be available to most, if not all, users. Public volumes contain files provided by DIGITAL, such as the operating system, help files, and diagnostic programs. Public volumes also contain files that system managers and users make available for general use. As long as file protection permits it, all users have access to public volumes and to the files on them.

Successful maintenance results from balancing users' needs for disk storage and the system's available storage. You must determine how mass storage devices on your system will be configured, which devices will hold public volumes, which devices will be available for users' private volumes, and how the public volumes will be configured.

Generally, there are two types of files on public volumes: system files and user files. System files refer to the operating system and its related files, plus common tools such as editors, compilers, and linkers. Public volumes contain the following kinds of files supplied by DIGITAL:

- The operating system in executable form and files related to the operating system
- Utility programs in executable form
- Diagnostic and test programs in executable form and files related to these programs
- Various system libraries such as macro libraries, object module libraries, shared run-time libraries, and error message libraries
- Text files such as help files
- Optional software in executable form plus related libraries and other files

In addition to these system files, you include on public volumes any files unique to an installation that must be accessible to many or all users. You also include all top-level user directories.

If you have a relatively small mass storage configuration, you will have both system files and user files on the same public volume. You should allocate disk space carefully, in such a case, and set disk quotas to ensure that user files do not exhaust the free space on the disk volume. (Entering the command SET DIRECTORY/VERSION_LIMIT=n is one way to control the amount of disk space consumed.)

In large mass storage configurations, you should keep all system files on one disk volume (known henceforth as a system disk or a system volume), and keep all user files on separate volumes. The system disk will be kept active reading system images, paging and swapping, spooling files, maintaining system logs, and so forth. See the *Guide to Setting Up a VMS System* for a complete list of the files that belong on the system disk; see also the SYSGEN commands CREATE and INSTALL in the *VMS System Generation Utility Manual*.

The most common arrangement is to have one public volume with system files and the directories of privileged users, and other public volumes dedicated to general user directories, databases, and applications required by your site.

Whichever arrangement you select, plan each public volume and test disk performance once the system is running. With the system disk, be sure that it has sufficient space for the VMS operating system to boot and accept updates, and that any user directories on the system disk are carefully controlled in size. Once the system is running, use the MONITOR Utility to analyze disk use and determine whether or not disk I/O is balanced across the configuration. Often it is possible to move system files off the system disk and use search lists or logical names to access them. In large configurations, secondary paging and swapping files can easily be placed on other devices to balance disk load. The *Guide to VMS Performance Management* provides detailed information on redistributing system files and achieving a balanced disk load.

2.1.1 Establishing Volume-Level Protection

You can apply file protection to disk volumes as well as to the files and directories on them. Even though users may have access to the directories and files on a volume, they cannot access any part of the volume without the proper volume protection code.

When you initialize or mount a public disk volume, user identification code (UIC) protection is set by default to full access, giving all users READ, WRITE, EXECUTE, and DELETE (RWED) access to the volume. You can use the /PROTECTION qualifier with the MOUNT or INITIALIZE command to specify a unique protection code. Once the volume is mounted, you can change the protection code with the SET VOLUME/PROTECTION command; for example:

```
$ SET VOLUME/PROTECTION=(S:RWED,O:RWED,G:RE,W:RE) DBA0:
```

Use the SHOW DEVICES command to verify the new protection code:

```
$ SHOW DEVICES/FULL DBA0:
  Disk $11$DBA0: ...       Dev Prot     S:RWED,O:RWED,G:RE,W:RE
```

Although access control lists (ACLs) are not typically set for disk volumes because of the performance expense, you can set ACLs on volumes for additional access control. For a full explanation of how to set UIC and ACL protection, see the *VMS DCL Concepts Manual* or the *Guide to VMS System Security*.

2.1.2 Using Volume Sets

A volume set is a collection of disk volumes bound into a single entity by the DCL command MOUNT/BIND. A volume set looks like a single, large volume. Files are automatically allocated anywhere on the volume set that space is available, disk quotas are enforced over the entire set, and a single directory structure covers the whole volume set. To provide a large homogeneous public file space, use a volume set.

To create files that are larger than a single physical disk volume, you must use a volume set. Note that the file system attempts to balance the load on the volume sets with tactics such as creating new files on the volume that is the least full at the time.

If you want several distinct areas of file storage, with different user bases or different management policies, you must use a separate volume (or volume set) for each area. For example, you might want one volume for permanent user storage, with limited disk quotas and regular backups. You might want another volume for "scratch" use, which has liberal or no quotas, is not backed up, and whose files are purged on a periodic basis. Each separate volume or volume set must contain a top-level user file directory for each user who will keep files on that volume.

An advantage of using separate volumes is their modularity. If one of the drives holding a volume set is out of service, the whole volume set will be unavailable because of its interconnected directory structure. When a drive holding a single volume is not functioning, only a well-defined set of files becomes unavailable.

When planning, keep in mind the following:

- Any single volume can be turned into a volume set by binding it with a newly initialized volume. Likewise, you can always add another newly initialized volume to an existing volume set.
- You can bind disk volumes of different types into the same volume set as long as it is not a system disk.
- You cannot bind two existing separate volumes containing files into a volume set. (The MOUNT command appears to let you do this, but the result is not a coherent volume set.)
- Enter the MOUNT/BIND command only once to bind a volume-set; thereafter, the volume-set association is recorded on the volumes. (See the *VMS Mount Utility Manual* for more information.)
- Once you have bound two or more volumes into a volume set, they cannot be separated. The only way to separate a volume set is to copy selectively sets of directories using the Backup Utility (BACKUP). (See the *VMS Backup Utility Manual* for more information.)

Caution: **Do not make the system disk part of a volume set because updates, upgrades, and optional product installations will not install correctly, and the VMS operating system will no longer boot successfully.**

2.2 Formatting Disks

Disks purchased from DIGITAL are preformatted by a field service diagnostic utility program. However, you may need or want to format a disk. Disks must be reformatted if they have been exposed to X-rays, degaussing (demagnetizing), or certain kinds of power disruptions. Also, you might want to format a disk if there are excessive parity errors. In such cases, you should contact your Field Service representative for assistance.

2.3 Initializing Public Volumes

The purpose of initializing a disk volume is to delete all old information from the volume and to impart to the volume a Files–11 structure recognized by the VMS operating system. This structure prepares a volume to receive data and stores it so that the operating system can locate it easily.

When initializing a public volume, you must specify the /SYSTEM qualifier with the DCL command INITIALIZE. For example:

```
$ INITIALIZE/SYSTEM device name[:] volume-label
```

You may want to use one or more of the following qualifiers when initializing a public volume:

- /CLUSTER_SIZE=n
- /HEADERS=n
- /INDEX=position
- /MAXIMUM_FILES=n
- /WINDOWS=n

Selecting appropriate values for these qualifiers and selecting the appropriate position for the index file involve making tradeoffs. Use the following guidelines for initializing public volumes. Refer to the *VMS DCL Dictionary* for complete information.

/CLUSTER_SIZE Qualifier

The /CLUSTER_SIZE qualifier specifies the number of disk blocks to be allocated each time disk space is needed. This is the minimum amount allocated whenever a file is created. When you select a small value (for example, 1) for /CLUSTER_SIZE, you may save disk space compared to selecting a relatively large value (for example, 4), because you will reduce the wasted space at the end of files. However, a small value may also result in increased disk and file fragmentation.

/HEADERS Qualifier

The /HEADERS qualifier specifies the number of file entries (called file headers) that you expect to have in the index file. The index file is a file holding the addresses of all disk files. The system accesses this index file each time it needs to locate a file on disk. An accurate estimate of the index file size allows the system to build one contiguous file, which translates into better system performance. Specifying too large a value wastes space.

If your estimate is too low and the index file needs to expand, the system uses a different part of the disk. However, a fragmented index file takes more time to scan. Do not be too generous with the /HEADERS value, because space allocated to headers cannot later be made available for file storage. A small disk might allocate 2000 entries in the index file, while a large disk might allocate the following:

```
$ INITIALIZE/HEADERS=100000 DUA3:
```

/INDEX Qualifier

The /INDEX qualifier determines the location of the index file on a volume, using the keyword BEGINNING, MIDDLE, or END. The index file lists the names and addresses of all disk files, so it is constantly referenced. The default position, MIDDLE, uses the smallest amount of head motion during most file processing if the disk is full. However, the BEGINNING keyword should be used if the disk is to contain either one or very few large, contiguous files. In rare instances, an application may require the use of the beginning of a disk; in this instance, use the END keyword.

When the Backup Utility copies a volume as the result of a BACKUP/IMAGE operation, it preserves the placement of the index file, if the output device is the same type. Otherwise, it defaults to MIDDLE.

/MAXIMUM_FILES Qualifier

The /MAXIMUM_FILES qualifier specifies the maximum number of entries in the index file, and therefore limits the number of files that a volume can contain. Once set, the maximum number of files for a volume cannot be increased without reinitializing the disk. Note that each directory and each extension header of a multiheader file counts as a file against this maximum value. For example, a small disk might have the following characteristic:

```
$ INITIALIZE/MAXIMUM_FILES=20000 DUA3:
```

/WINDOWS Qualifier

The /WINDOWS qualifier sets the default number of mapping pointers to be allocated for file windows. When a file is opened, the file system uses mapping pointers to access data in the file. The file system can read one file segment into memory for each available pointer. For a large disk of 500 Mbyte, you may cite a large number of pointers, for example:

```
$ INITIALIZE/WINDOWS=10 DUA3:
```

2.4 Mounting Public Volumes

Mounting a volume or volume set establishes a relationship between the volume or volume set and the device on which the volume is physically loaded. When mounting a volume set, first physically mount all disks and put them online. Then enter the MOUNT command. Once invoked, the Mount Utility allocates the device, checks to see that the device is correctly loaded, and reads and verifies the disk label.

To mount a public volume, include the /SYSTEM qualifier with the DCL command MOUNT:

```
$ MOUNT/SYSTEM device-name volume-label logical-name
```

You can modify the command with any of the following qualifiers:

- /ACCESSED=n
- /BIND=volumeset_name
- /CLUSTER
- /COMMENT=text
- /EXTENSION=n
- /[NO]ASSIST
- /[NO]MOUNT_VERIFICATION
- /SHARE
- /WINDOWS=n

These qualifiers are described in the following section. For a complete description of the command and its qualifiers, see the *VMS Mount Utility Manual.*

/ACCESSED Qualifier

The /ACCESSED qualifier specifies the number of directories that the system will keep in memory for ready access. The file system selects the most recently used directories and, by storing them in memory, substantially reduces the overhead associated in directory operations. (For volumes mounted with the /SYSTEM qualifier, the SYSGEN parameter ACP_DINDXCACHE overrides this value.)

When you create a volume set, specify reasonable values for the /ACCESSED qualifier on each volume because the total number of directory file control blocks retained will be the sum of the values of all the /ACCESSED qualifiers specified for the volume set.

For example, on a large 500 Mbyte disk you might select a value of 40, but on a small disk you might specify the following:

```
$ MOUNT/ACCESSED=2 DUA3:
```

/BIND Qualifier

The /BIND qualifier creates a volume set of one or more disk volumes or adds an existing volume to a volume set. Include a volume set name as part of the qualifier, for example:

```
$ MOUNT/SYSTEM/BIND=CLIENTS DMAO:,DMA1: EUROPE,ASIA
```

/CLUSTER Qualifier

The /CLUSTER qualifier specifies that after a public volume is successfully mounted on the local node with the /SYSTEM qualifier, it is to be mounted on every other node in the existing VAXcluster, for example:

```
$ MOUNT/SYSTEM/CLUSTER DBA1: SALES_86
```

For more information on mounting clusterwide disk volumes, see the *VMS VAXcluster Manual.*

/COMMENT Qualifier

The /COMMENT qualifier includes a quoted text string that you specify as part of the mount request. This qualifier is useful in situations where operator assistance is expected, because it passes on information such as the physical location of a particular volume that is required, for example:

```
$ MOUNT/SYSTEM DYA1: SALES_86/COMMENT="Volume in Rack 2."
```

You should encourage users to take advantage of this feature.

/EXTENSION Qualifier

The /EXTENSION qualifier specifies the number of disk blocks allocated to a file each time it is expanded. The cluster size sets the initial disk block allocation and the extension qualifier determines how the file grows. The quantity by which a file is extended is always a multiple of the cluster size for the volume, because the cluster size is the minimum allocation quantity in that volume. In the following example, a small disk with a cluster size of 1 disk block, selects an extension size of 2 disk blocks:

```
$ MOUNT/EXTENSION=2 DUA3:
```

/[NO]ASSIST Qualifier

The /NOASSIST qualifier disables automatic notification of any mount failures. Normally, an operator is told whenever the system is unable to process a mount request. This gives the operator an opportunity to terminate the operation or correct the problem so the mount can continue. Encourage users to take advantage of this feature, which repeatedly alerts the operator of a mount request until the request is satisfied. For example, to disable operator-assisted mounts, you could enter the following command:

```
$ MOUNT/SYSTEM/NOASSIST DBA1: SALES_86
```

The /ASSIST qualifier is the default except during system startup. During the execution of STARTUP.COM, operator-assisted mounts are disabled because there is no operator defined at this time.

/[NO]MOUNT_VERIFICATION Qualifier

The /[NO]MOUNT_VERIFICATION qualifier enables or disables the mount verification feature on disks and magnetic tapes. By default, the mount verification feature is enabled. If a device goes offline or becomes write-locked, mount verification notifies the operator of the error condition, and then checks to see that the volume identification codes recorded before and after the error condition are identical. Once verified, the operations can continue from the error point, rather than restarting the mount from the beginning. Mount verification retains pending requests in a queue while attempting to recover, instead of aborting them. To disable mount verification, you could enter the following command:

```
$ MOUNT/SYSTEM/NOMOUNT_VERIFICATION MTB1: ACCOUNTS_DUE
```

/SHARE Qualifier

The /SHARE qualifier specifies that other users can access the current volume. Use this qualifier instead of the /SYSTEM qualifier when you have READ access to a volume, but not the SYSNAM privilege that /SYSTEM requires. While the qualifier creates a volume that can be shared by several users, the only true public volume is made by /SYSTEM. Use /SHARE in the following way:

```
$ MOUNT/SHARE DLAO: COST_ACCOUNT
```

/WINDOWS Qualifier

The /WINDOWS qualifier on the MOUNT command resets the number of mapping pointers to be allocated for file windows. The default numbers of windows is set with the INITIALIZE command. When a file is opened, the file system uses mapping pointers to access data in the file. The file system can read one file segment into memory for each retrieval pointer. For a small disk you may cite a modest number of pointers; for example:

```
$ MOUNT/WINDOWS=4 DUA3:
```

Upon successful completion of the operation, MOUNT notifies you with a message sent to SYS$OUTPUT. If the operation fails for any reason, MOUNT notifies you with an error message.

2.5 Using the Exchange Utility to Transfer Information

This section describes how to use the Exchange Utility to transfer information between foreign volumes and VMS Files–11 volumes. The Exchange Utility (EXCHANGE) converts the format of files, as appropriate, when transferring files between volumes of different structures. EXCHANGE recognizes all Files–11 and RT–11 disk volumes on VMS devices, as well as all DOS–11 and RT–11 formatted volumes on 9-track magnetic tape devices.

Use EXCHANGE commands to perform any of the following tasks:

- Locate bad blocks on volumes
- List directories of volumes
- Transfer files to and from volumes
- Delete and rename files for block-addressable devices
- Create foreign volumes
- Manipulate Files–11 files that are images of foreign volumes

For information on how to use EXCHANGE and for a listing of all the commands, qualifiers, and parameters, see the *VMS Exchange Utility Manual*.

2.5.1 Running the Exchange Utility Interactively

To invoke the EXCHANGE Utility, enter the following command in response to the DCL prompt:

```
$ EXCHANGE
```

The utility displays the following prompt:

```
EXCHANGE>
```

You can now enter any EXCHANGE command; for example:

```
EXCHANGE> DIRECTORY MFAO:/VOLUME=DOS11/FULL
```

This command lists all the files on the DOS–11 magnetic tape mounted on MFA0:. In this case, the magnetic tape will be rewound before the files are listed.

The following example illustrates the use of the MOUNT command within the EXCHANGE Utility. This command mounts the foreign volume that is loaded in the RK07 device DMA1:, making the volume available for subsequent commands. In this case, EXCHANGE recognizes that the volume itself is write-locked, and displays the following message:

```
EXCHANGE> MOUNT DMA1:
%EXCHANGE-I-WRITELOCK, volume is write-locked
%EXCHANGE-I-MOUNTED, volume DMA1: mounted
```

To exit from EXCHANGE and return to DCL level, use the EXCHANGE command EXIT or CTRL/Z.

You can also type EXCHANGE at the DCL prompt and enter a single EXCHANGE command on the same line. EXCHANGE executes the single command and returns to the DCL prompt. For example, you can list the directory of a foreign volume by appending the EXCHANGE command DIRECTORY to the DCL command EXCHANGE, as follows:

```
$ EXCHANGE DIRECTORY DMA1:/VOLUME_FORMAT=RT11
```

This DCL command lists the directory of the RT–11 volume mounted (/FOREIGN) on DMA1.

Tasks requiring more than one EXCHANGE command cannot be performed at DCL level unless you design a command procedure. Then you execute the command procedure at the DCL command level, as the following section describes.

2.5.2 Using a Command Procedure to Exchange Information

The command procedure in this example exchanges files between the console device and the current directory on disk. This procedure assumes that the files to be copied are in standard format, as determined by file type.

```
$ WRITE SYS$OUTPUT " Command file to copy files to/from the system"
$ WRITE SYS$OUTPUT " console storage medium and the current directory."
$ WRITE SYS$OUTPUT " "
$ INQUIRE MOUNT "Is system console storage medium mounted (Y/N)?"
$ IF MOUNT THEN GOTO MOUNTED
$ WRITE SYS$OUTPUT "Please place the system console medium in the console drive"
$ INQUIRE MOUNT "and type RET when ready"
$ RUN SYS$SYSTEM:SYSGEN
CONNECT CONSOLE
$ MOUNT/SYSTEM/FOREIGN CSA1: "VAX console"
$
$ MOUNTED:
$ INQUIRE DIR "Copy from console medium (Y/N)?"
$ IF DIR THEN GOTO FROMCON
$ INQUIRE SOURCE "Enter file name(s)"
$ IF SOURCE .EQS. "" THEN GOTO EXIT
$ EXCHANGE COPY /LOG 'SOURCE' CSA1:
$ GOTO EXIT
$
$ FROMCON:
$ INQUIRE SOURCE "Enter console file name"
$ IF SOURCE .EQS. "" THEN GOTO EXIT
$ EXCHANGE COPY /LOG CSA1:'SOURCE' *
$ EXIT:
$ DISMOUNT CSA1:
$ MOUNT/SYSTEM/FOREIGN/NOWRITE CSA1: "VAX console"
```

3 Performing Disk and Magnetic Tape Operations

This chapter describes the following disk and magnetic tape operations:

- Using the Operator Communication Manager (OPCOM)
- Servicing mount requests from users
- Servicing requests from the Backup Utility
- Maintaining volume integrity
- Performing mount verification

3.1 Using the Operator Communication Manager (OPCOM)

The Operator Communication Manager (OPCOM) is an online system management tool that does the following:

- Handles requests from users
- Delivers requests to the operator
- Records messages in the operator log file

The following sections describe how to set up an operator terminal to enable OPCOM and how to handle user requests.

3.1.1 Enabling the Operator Terminal

When you respond to system users requests to mount disk and magnetic tapes, you work from a terminal that has been defined as an operator terminal. Normally, the console terminal is designated as the operator terminal (OPA0:).

To designate a particular terminal as an operator terminal, enter the DCL command REPLY/ENABLE from the desired terminal. When OPCOM receives a REPLY/ENABLE command, it displays the following message on the enabled operator terminal:

```
%%%%%%%%%%%  OPCOM, dd-mmm-yyyy hh:mm:ss.cc, %%%%%%%%%%%
Operator _nodename$terminal-name: has been enabled, username USERNAME
```

OPCOM writes the same message to the operator's log file. This message tells which terminal has been established as an operator terminal and when it was established.

To disable operator terminal functions, log out or enter the DCL command REPLY/DISABLE.

3.1.2 Restarting OPCOM

If, under abnormal conditions, OPCOM is deleted or suspended, you can restart it manually. To restart OPCOM, log in to the system manager's account and execute STARTUP.COM by entering the following command:

```
$ @SYS$SYSTEM:STARTUP OPCOM
```

If the system crashes, OPCOM normally restarts automatically as soon as the system startup procedure is executed. However, as previously mentioned, OPCOM is not enabled by default at system startup time on workstation processors. If you enable OPCOM on a workstation processor, the console prompt appears in the OPCOM window when the system crashes.

3.1.3 Requests for Operator Assistance

Users communicate with the operator using the REQUEST command, specifying either the /TO or the /REPLY qualifier. The REQUEST command sends a text message to the operator, but it does not initiate any further system action.

The REQUEST command with the /REPLY qualifier means that the user is waiting for a response from the operator. The user's current process is put in a wait state until you respond. When the user enters a REQUEST/REPLY command, the request appears at the operator terminal in the following format:

```
%%%%%%%%%%  OPCOM,dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
request request-id from user USERNAME
__terminal-name:, "message-text"
```

This message explains which user sent the message, the time it was sent, the request identification number assigned to the message, the originating terminal, and the request message.

If the user enters a REQUEST/TO command, the request appears at the operator terminal in a format similar to the one above, but without a request identification number. The message will have the following format:

```
%%%%%%%%%%%  OPCOM,dd-mmm-yyyy hh:mm:ss.cc  %%%%%%%%%%%
request from user USERNAME
__terminal-name:, "message-text"
```

If all operator terminals are disabled when a user enters a REQUEST/REPLY command, OPCOM returns a message to the user indicating that no operator coverage is available. However, it records this request and all subsequent requests in the operator log file (SYS$MANAGER:OPERATOR.LOG). For more information on the operator log file, see Chapter 7.

Users can also enter the MOUNT command for a disk or magnetic tape operation. (See Section 3.2.) For more detailed information on the REQUEST and the MOUNT commands, including a list of all the command qualifiers, see the *VMS DCL Dictionary* and the *VMS Mount Utility Manual.*

3.1.4 Responding to User Requests

You communicate with system users by entering the REPLY command, in one of the following forms:

- Use the **REPLY/ABORT=identification-number "message-text"** command to indicate that the user request is canceled.
- Use the **REPLY/PENDING=identification-number "message-text"** command when the request has been put in a wait state until it can be completed. This command implies that the originating request was either a REQUEST/REPLY or a MOUNT command. The user cannot enter other commands until the operator fulfills or aborts the request.
- Use the **REPLY/TO=identification-number "message-text"** command when the request is fulfilled.

In situations where users submit a MOUNT/ASSIST command and the desired device is unavailable, you can substitute another device. Load the requested volume on the alternate device and ready the device before entering the REPLY command in the following form.

```
REPLY/TO=identification-number "SUBSTITUTE  device-name"
```

You can abbreviate the word SUBSTITUTE to **S** and can use uppercase or lowercase characters. The following example shows how an operator redirects the mount operation to DMA1:

```
$ REPLY/TO=24 "SUBSTITUTE  DMA1"
```

3.2 Handling Requests for Mounting Volumes

At many installations, operators perform the physical mounting and dismounting of public and private disk and magnetic tape volumes. At these installations, users do not have access to the computer room so they communicate with you using DCL commands. When users need to have a tape or disk mounted, they submit a request using the REQUEST command or the MOUNT command. The REQUEST command would request assistance and describe the volume to be mounted, whereas the MOUNT command specifies the name of the device to be allocated, the volume to be mounted, and perhaps a comment indicating the location of the volume and its jacket label. For example,

```
$ MOUNT DYAO: ACCOUNTS_USA/COMMENT="Write enable, volume in rack 1"
```

Use the following steps to respond to a user request for mounting disk or magnetic tape media:

1. Locate the designated volume. If the user intends to write to a disk or tape, toggle the disk drive's write-lock switch or insert a write ring in the tape volume. (Any write operation will fail if the volume is not write-enabled.)
2. Place the volume on the specified device and on disk drives, press the START or RUN button; on magnetic tape drives, press the LOAD button.
3. Enter the REPLY command with the appropriate qualifier.

If a user wants a particular disk to be initialized but has already entered the MOUNT command, dismount the volume with the DCL command DISMOUNT, enter the DCL command INITIALIZE, and then mount the volume a second time, using the MOUNT command. Be careful to specify the user's ownership when you remount the device.

3.2.1 Mounting Disks and Single Magnetic Tape Volumes

When a user requests that a specific disk or magnetic tape be mounted on a device, the following type of message appears on the operator terminal:

```
%%%%%%%%%%%  OPCOM  dd:mmm:yyyy:hh:mm:ss.cc  %%%%%%%%%%%
request request-id, from user USERNAME
```

For example, a user requesting to mount the volume TEST_FILES on the device DMA2 could enter the following command:

```
$ MOUNT DMA2: TEST_FILES/COMMENT="Shelf slot 6B"
```

OPCOM notifies you of the request by displaying a message similar to the following at the operator terminal:

```
%%%%%%%%%%%  OPCOM, 28-DEC-1988 15:47:50.26  %%%%%%%%%%%
request 5, from user MALCOLM
Please mount volume TEST_FILES in device _DMA2:
Shelf slot 6B
```

Once you receive the request, OPCOM delivers a confirmation to the user, in a format similar to the following:

```
%MOUNT-I-OPRQST, Please mount volume TEST_FILES in device _DMA2:
Shelf slot 6B
```

After you locate the physical volume and place it on the device, OPCOM notifies the user that the volume is on the device and that the task is complete.

```
%MOUNT-I-MOUNTED, TEST_FILES mounted on _DMA2:
%MOUNT-I-RQSTDON, operator request canceled
-- mount completed successfully.
```

Instead of requesting a specific hardware device, such as DMA2 or MTA0 for mounting a volume, the user can make a generic MOUNT request. A generic mount request specifies a type of device and lets the system find an available device in that class. For example, to mount the volume CITIES on any magnetic tape drive whose name begins with MT, the user would enter the following command:

```
$ MOUNT MT:  CITIES/COMMENT="Slot 12c"
```

If the user has already allocated a drive whose name begins with MT, the VMS Mount Utility requests that the operator mount CITIES on that particular drive. If no device has been allocated, the MOUNT utility allocates the first available MT tape drive it finds and requests you to mount CITIES on that drive.

3.2.2 Mounting Magnetic Tape Volume Sets

The procedure for mounting a magnetic tape volume set is similar to the procedure for mounting a single magnetic tape volume described in Section 3.2.1. When mounting a volume set, however, make sure that all the volumes in the set contain write rings if the user intends to write to any one of the volumes in the set. If even one of the volumes in the set does not contain a write ring at mount time, all volumes will be write-locked; the system will be unable to write to any of them. You can load the volumes on the drives that have been allocated and place the drives on line.

The next two sections describe how to handle user requests to mount a magnetic tape volume set under the following conditions:

- Automatic volume switching is enabled by default
- Automatic volume switching is disabled, with the MOUNT/NOAUTOMATIC command

3.2.2.1 Mounting Volume Sets with Automatic Switching

The VMS operating system supports automatic volume recognition (AVR) and automatic volume labeling (AVL) for ANSI magnetic tapes. If the magnetic tape volume is mounted with AVR and AVL enabled, which it is by default, then your only task is to physically place the next volume in the drive. The system automatically reads the label and mounts the volume. The system then sends a message to the operator terminal informing you that a volume switch has occurred.

With automatic volume switching, you can load each successive magnetic tape in the volume set on one of the allocated drives any time before the magnetic tape being processed reaches an end-of-tape position. The magnetic tape file system mounts or initializes (or both mounts and initializes) the next magnetic tape, and then notifies you that a switch has occurred. The magnetic tape file system switches volumes by issuing pending read or write requests to the next volume in the set.

Note that to take advantage of the automatic volume-switching capability, more than one magnetic tape drive must be allocated to the magnetic tape volume set; otherwise, automatic volume switching will be implicitly disabled.

3.2.2.2 Mounting Volume Sets without Automatic Switching

This section describes how to handle requests for mounting continuation volumes in a volume set when automatic volume switching is disabled or when the magnetic tape file system cannot mount a given volume.

When a user is reading or writing to a magnetic tape and that tape reaches an end-of-tape condition, the system suspends processing and sends a request to you to mount the next magnetic tape in the volume set. For example:

```
%%%%%%%%%%%  OPCOM, 28-DEC-1988 15:23:31.78  %%%%%%%%%%%
request 3, from user PLAW
MOUNT new relative volume 2 (DWOQT2) on MTA1:
```

The user does not see this message and may not realize that another tape is needed to complete the read or write operation.

After loading the continuation volume on the drive specified in the mount request, you mount the volume by entering the REPLY command with one of three qualifiers:

- Use the **REPLY/TO=request_id [volume identifier]** command for both read and write operations. During a write operation, you use the /TO qualifier if you want the volume identifier that is specified in the mount request to be written on the continuation volume.

  For example, to respond to the mount request 3, you would mount volume DW0QT2 on drive MTA1:, and enter the following command:

  ```
  $ REPLY/TO=3 "DWOQT2"
  ```

  The volume identifier is required in certain instances, but not in others. See the explanation that follows this list.

- Use the **REPLY/INITIALIZE_TAPE** command for write operations if the volume identifier on the continuation volume does not match the one specified in the mount request. The file system reinitializes the tape and mounts the volume with the new volume identifier. The magnetic tape file system then performs access checks and initializes the volume as if the INITIALIZE command had been specified, for example:

  ```
  $ REPLY/INITIALIZE_TAPE=3 "DWOQT2"
  ```

- Use the **REPLY/BLANK_TAPE** command to write to an unformatted volume. This qualifier initializes the volume and requires the VOLPRO and OPER privileges to avoid a runaway tape or timeout condition. Either of the following REPLY commands is valid:

  ```
  $ REPLY/BLANK_TAPE=3
  ```

  ```
  $ REPLY/BLANK_TAPE=3 "DWOQT2"
  ```

Specifying the volume identifier in either the MOUNT command or the REPLY/TO command is essential during write operations because it ensures that the correct volume is mounted on the drive and links the continuation volume to the volume set.

You can omit the volume identifier with the REPLY/TO command under two circumstances. When reading from tape, the volume identifier is optional. During a write operation, you must omit the volume identifier to preserve the accessibility code on a volume. If you initialize and mount a volume set in which each volume has a unique accessibility character that you want to maintain, avoid using the volume identifier because it causes the accessibility character of the first volume in the set to overwrite the accessibility code on the continuation volume. For example, to preserve the accessibility character you would enter the following command, where 3 is the request identification number:

```
$ REPLY/TO=3
```

Once it receives the REPLY command, the magnetic tape file system performs checks on the continuation volume to ensure that it is the correct volume. As long as it is the correct volume with proper access codes, the system mounts the volume and reissues pending read or write requests to the continuation volume. If the volume fails any of these access checks, the volume is not mounted (or initialized and mounted, in the case of a blank tape).

3.3 Handling Requests from the Backup Utility

When a backup operation is submitted as a batch job, operators receive requests to load the next volume of a save set or to reload a volume in the event of an error. (These requests go to the user when the Backup Utility is invoked interactively.)

3.3.1 Writing to a Save Set

During a backup SAVE operation, while the file system is copying data from disk to magnetic tape, you receive a message if an operation requires the loading of an additional volume. The message states the date and time, a request number, the user name, and the device name, as shown in the following example:

```
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:32.31 %%%%%%%%%%%
Request 24, from user TOM.
%BACKUP-I-READYWRITE, mount volume 2 on _MTA0: for writing
```

To continue the backup operation, load a scratch volume, ready the device, and enter a REPLY/TO=n command, specifying the request number as parameter *n* as shown in the following example:

```
$ REPLY/TO=24

%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:34.14 %%%%%%%%%%%
Request 24 completed by operator OPA0.
```

You can also abort the backup operation by typing a REPLY/ABORT command, as shown in the following example:

```
$ REPLY/ABORT=24

%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:34.14 %%%%%%%%%%%
Request 24 aborted by operator OPA0.
```

3.3.2 Reading from a Save Set

During a restore operation, when the file system reads data from backup media and writes it to disk or tape, you receive a message if a restore operation requires the loading of an additional volume. The message states the date and time, a request number, the user name, and the device name, as shown in the following example:

```
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:32.31 %%%%%%%%%%%
Request 24, from user TOM.
%BACKUP-I-READYREAD, mount volume 2 on _MTA0: for reading
```

To continue the restore operation, place the next volume of the save set on the drive, ready the device, and enter a REPLY/TO command, as shown in the following example:

```
$ REPLY/TO=24
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:34.14 %%%%%%%%%%%
Request 24 completed by operator OPA0.
```

You can also abort the restore operation by entering a REPLY/ABORT command, as shown in the following example:

```
$ REPLY/ABORT=24
```

```
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:34.14 %%%%%%%%%%%
Request 24 aborted by operator OPAO.
```

3.3.3 Recovering from an Error

On certain BACKUP save or restore errors, you receive messages stating the date and time, a request number, the user name, the device name, and reply options. On read errors from the save set, the options are usually CONTINUE and QUIT. On write errors to the save set, the options are usually RESTART and QUIT. You should take one of the following actions:

- If you can fix the problem and are given the CONTINUE option, ready the device, and enter a REPLY/TO=n command specifying the request number in parameter *n* and the word CONTINUE as the text; for example,

  ```
  REPLY/TO=n "CONTINUE"
  ```

- If you can fix the problem and are given the RESTART option, load the volume (if necessary), ready the device, and enter a REPLY/TO=n command specifying the request number and the word RESTART as the text; for example,

  ```
  REPLY/TO=n "RESTART"
  ```

- If you cannot fix the problem, enter a REPLY/TO=n command specifying the request number and the word QUIT as the text; for example,

  ```
  REPLY/TO=n "QUIT"
  ```

The text in the REPLY/TO command can be uppercase or lowercase, and can be abbreviated to the first character.

Assume, for example, that you have mounted a tape without mount verification enabled and you receive the following messages during a restore operation (when the save set is being read):

```
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:30.58 %%%%%%%%%%%
Message from user RESJOB.
%BACKUP-E-FATALERR, fatal error on MT:[]SAVE.;
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:30.89 %%%%%%%%%%%
Message from user RESJOB.
-SYSTEM-F-MEDOFL, medium is offline
%%%%%%%%%% %OPCOM,28-DEC-1988 17:02:31.05 %%%%%%%%%%%
Request 24, from user RESJOB.
%BACKUP-I-SPECIFY, specify option (QUIT or CONTINUE)
```

If you check the drive and find that it lost its vacuum, ready the tape drive and enter the following command:

```
$ REPLY/TO=24 "CONTINUE"
CONTINUE
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:33.41 %%%%%%%%%%%
Request 24 completed by operator OPAO.
```

If you find the drive inoperable, enter the QUIT reply:

```
$ REPLY/TO=24 "QUIT"
QUIT
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:33.41 %%%%%%%%%%%
Request 24 completed by operator OPA0.
```

If the error occurs during a save operation (when the save tape is being written), you have a choice of QUIT or RESTART:

```
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:30.58 %%%%%%%%%%%
Message from user RESJOB.
%BACKUP-E-FATALERR, fatal error on MT:[]SAVE.;
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:30.89 %%%%%%%%%%%
Message from user RESJOB.
-SYSTEM-F-MEDOFL, medium is offline
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:31.05 %%%%%%%%%%%
Request 24, from user RESJOB.
%BACKUP-I-SPECIFY, specify option (QUIT or RESTART)

(You ready the magnetic tape drive.)
   .
   .
   .
$ REPLY/TO=24 "RESTART"
RESTART
%%%%%%%%%%% OPCOM,28-DEC-1988 17:02:33.35 %%%%%%%%%%%
Request 24 completed by operator OPA0.
```

The RESTART option permits you to restart a multivolume backup operation at the beginning of the current volume. If you specify the QUIT option, you must restart the backup operation completely.

3.4 Maintaining Volume Integrity

To enhance performance, the system caches in memory information about a disk volume's free space, file identifications, quota file entries, and file headers. You determine the degree of caching with the ACP cache system parameters, discussed in the *VMS System Generation Utility Manual*. Individual users can alter cache sizes on their volumes with qualifiers of the DCL command MOUNT, described in the *VMS Mount Utility Manual*.

The system writes the information in the caches to the disk when the disk is dismounted or the system is shut down. Removal of a disk before the caches are written to disk loses any changes made to the information in the caches. Therefore, neither you nor the user should do the following:

- Write-lock a volume while it is mounted
- Remove a volume from a drive until it has been dismounted
- Halt the system without performing an orderly shutdown procedure (see *Guide to Setting Up a VMS System*)

If anyone write-locks a volume at mount time, the system additionally applies a software write-lock. To write enable a volume that was mounted while the **WRITE LOCK** switch was on, you must first dismount the volume, write enable the drive, and then remount the volume. If a volume was mounted on a drive with write-lock off and someone toggles the **WRITE LOCK** switch while mount verification is enabled for the volume, the volume enters mount verification. All I/O operations to the volume are suspended.

Section 3.5.2 describes how recovery is affected with write-lock mount verification. Without the mount verification facility, you would have to dismount the volume, write enable the drive, and then remount the volume.

At mount time, if the system detects that the caches were not written back the last time the volume was used, the system automatically rebuilds the file information by scanning the contents of the volume. However, file headers for files open at the time of the improper dismount may be partially or entirely lost.

3.5 Mount Verification

Mount verification provides a recovery mechanism for disk and tape operations. If a device is write-locked or goes off line while mount verification is enabled, you can correct the problem and continue the operation, rather than abort the entire disk or tape operation. As soon as mount verification detects an error condition, it stalls all I/O to the disk or tape, waits for the error resolution, and then validates the volume on the enabled device.

The mount verification feature of disk and tape handling generally leaves users unaware that a mounted disk or tape has gone off line and returned on line, or in some other way has become unreachable and then restored. Mount verification is enabled by default with the /MOUNT_VERIFICATION qualifier when the disk or tape is mounted. To disable mount verification, you must specify /NOMOUNT_VERIFICATION when mounting the disk or tape. Note that this feature applies to ANSI mounted tapes, foreign mounted tapes, and Files–11 disks.

Mount verification sends two messages: one goes to OPCOM and the other, distinguished by the prefix %SYSTEM-I-MOUNTVER, goes directly to the system console (OPA0, bypassing OPCOM). The second message is a form of insurance in cases where OPCOM might be unavailable. For example, if the system disk undergoes mount verification or if OPCOM is not present on a system, you would at least receive the messages with the %SYSTEM-I-MOUNTVER prefix. Under normal circumstances, both messages are received at the operator's terminal, with the %SYSTEM-I-MOUNTVER message arriving first.

3.5.1 Mount Verification for Offline Devices

Any offline condition initiates mount verification. Usually an offline condition results from a hardware or user error—for example, someone inadvertently presses the offline button. Once a device is off line, the hardware (and for some disks, the software) marks the disk or tape as "invalid", and I/O requests fail. As long as mount verification is enabled, the software marks the volume to indicate that it is undergoing mount verification and stalls all I/O operations to the disk or tape until the problem is corrected. OPCOM then issues a message in the following format to the operators enabled for DISKS and DEVICES or TAPES and DEVICES to announce the disk or tape's unavailability:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Device 'device-name' is offline.
Mount verification in progress.
```

When a mounted disk or tape volume goes offline while mount verification is enabled, you can try to recover or you can terminate the mount request. The following options are available:

- Try to put the device on line by toggling the START or RUN button on disks or the LOAD button on tapes.
- If the disk or tape drive is faulty, but another functioning drive is available on the same controller, move the disk or tape to the functioning drive and swap the unit select plugs.
- Take the disk or tape out of the offline and verification pending state by shutting down mount verification with one of the three techniques described in Section 3.5.3. These techniques include canceling the mount request, dismounting the volume, and allowing mount verification to time out.

If the device is successfully put back on line, the mount verification software that polls the disk or tape drive will begin verification. The system checks to see that the currently mounted disk or tape has the same identification as the previously mounted volume. Thus, mount verification confirms that this is the same disk or tape as was previously mounted and no switching has occurred.

If the drive contains the wrong volume, OPCOM issues a message in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Device 'device-name' contains the wrong volume.
Mount verification in progress.
```

Once mount verification succeeds, the disk is marked as valid, and OPCOM issues the following message:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Mount verification completed for device 'device-name.'
```

At this point, I/O operations to the disk or tape proceed, as shown in the following example:

```
%%%%%%%%%%% OPCOM, 28-DEC-1988 11:54:54.12 %%%%%%%%%%%
Device DMA0: is offline.
Mount verification in progress.

%%%%%%%%%%% OPCOM, 28-DEC-1988 11:57:34.22 %%%%%%%%%%%
Mount verification completed for device DMA0:.
```

In this example, the message from OPCOM informs the operator that device DMA0 went offline and mount verification was initiated. The operator finds that the drive was accidentally powered down and successfully powers it up again. The next message indicates that mount verification is satisfied that the same volume is on the drive as was there before the error. All I/O operations to the volume resume.

3.5.2 Mount Verification for Write-Locked Devices

Devices become write-locked when a hardware or user error occurs while a disk or a magnetic tape volume is mounted for a write operation. For example, if a disk is write-locked or a tape is missing a write ring, the hardware generates an error. As soon as the software discovers that the disk or tape is write-locked (for example, when an I/O operation fails with a write-lock error), mount verification begins.

OPCOM issues a message to the operators enabled for DISKS and DEVICES or TAPES and DEVICES to announce the disk or tape's unavailability:

```
%%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Device 'device-name' has been write-locked.
Mount verification in progress.
```

You can either recover the operation or terminate mount verification. Your options include the following:

- Enable the drive for writing by toggling the disk or tape drive's hardware WRITE LOCK switch, or check to see that a tape volume has a write ring.
- If the disk or tape drive is faulty, but another functioning drive is available on the same controller, move the disk or tape to the functioning drive and swap the unit select plugs. (Note that switching to another drive causes the volume to undergo offline mount verification. Once that completes, the write-lock mount verification continues.)
- Terminate the mount operation by shutting down mount verification with one of the techniques described in Section 3.5.3. These techniques include canceling the mount request and dismounting the volume.

Once the mount verification software determines that the volume is in a writable state, I/O operations to the disk resume with no further messages.

3.5.3 Canceling Mount Verification

You can cancel a mount verification request in one of the following ways:

- Invoke a special canceling routine from the console terminal.
- Dismount the volume with the DCL command DISMOUNT from a process that is not hung.
- If the device is off line, allow mount verification to time out. The expiration value is 10 minutes by default; however, you can use the system parameter MVTIMEOUT (for disk) or TAPE_MVTIMEOUT (for magnetic tape) to set the value to whatever you want. When the time expires, the system automatically cancels the pending mount verification. Note that a mount verification initiated by a write-lock condition will not time out.

The following sections describe these methods in more detail.

3.5.3.1 Cancellation Commands

To cancel mount verification, you run a special interrupt program called Interrupt Priority C (IPC). The IPC issues a software interrupt to gain the attention of the VMS operating system. You use the IPC to cancel mount verification, enter the debugger, or recalculate the quorum on a cluster. (The debugger, in this case, refers to the system level debugger, XDELTA.)

To invoke the IPC, enter the following sequence of commands from the console terminal of your VAX processor:

```
CTRL/P
>>>HALT
>>>D/I 14 C
>>>CONT
IPC>
```

To exit from the IPC and resume normal operations, press CTRL/Z.

The IPC program converts lowercase characters to uppercase, issues the terminal bell character whenever it receives illegal characters (such as most control characters), compresses multiple spaces, and ignores leading spaces.

You can enter one of the following commands in response to the IPC> prompt:

- Use the **C device-name** command to cancel any pending mount verification on the device specified. (A warning is given if no mount verification was in progress for that device.) For example:

  ```
  IPC> C MTA1:
  ```

- Use the **X** command to transfer control to the debugging tool XDELTA (provided it was loaded with the system by setting the appropriate value in the bootstrap file). If XDELTA has not been loaded, the prompt IPC> is reissued. XDELTA, which is described in the *VMS Delta/XDelta Utility Manual,* may prove useful if you are debugging privileged software on a VAX-11/782 attached processor system. For example:

  ```
  IPC> X
  ```

- Use the **Q** command to recalculate the quorum on a cluster.

When a pending mount verification is canceled, OPCOM prints a message in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Mount verification aborted for device 'device-name.'
```

After you successfully cancel a pending mount verification with this technique, you must dismount and then remount the volume before you can access it again, as the following example demonstrates:

```
%%%%%%%%%%% OPCOM, 28-DEC-1988 10:54:54.12 %%%%%%%%%%%
        Device DBA0: is offline.
        Mount verification in progress.

CTRL/P
>>>HALT
>>>D/I 14 C
>>>CONT
IPC>C DBA0
IPC>^Z
%SYSTEM-I-MOUNTVER, _DBA0: has aborted mount verification.
%%%%%%%%%%% OPCOM, 28-DEC-1988 10:56:26.13 %%%%%%%%%%%
Mount verification aborted for device DBA0:
```

In this example, device DBA0 is off line, but you are unable to spin the disk back up. There is no other available drive on the controller, so it is not possible to switch the unit select plugs of the two drives. You do not enter a DISMOUNT command for the disk, because it was mounted as a private volume and you do not have access to it. The %SYSTEM-I-MOUNTVER message also appears here because this is the console terminal.

3.5.3.2 Dismounting the Volume

If it is possible for you to enter the DCL command DISMOUNT for the volume, then you can abort mount verification by dismounting the volume in question. (DISMOUNT requires GRPNAM and SYSNAM user privileges to dismount group and system volumes, respectively.) To dismount a volume, follow these steps:

1. Log in at another terminal or use any logged-in terminal that has access to the volume. It does not have to be an operator's terminal.
2. Enter the DISMOUNT/ABORT command for the volume.

   When you cancel a pending mount verification by dismounting the volume, OPCOM issues a message in the following format:

   ```
   %%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc. %%%%%%%%%%%%
   Mount verification aborted for device 'device-name.'
   ```

   If you do not have access to the volume, you will receive an error message. You can try again if you can find an appropriate process to use. If your process hangs, it is the system file ACP that is hung, and you cannot use this technique to cancel mount verification.
3. Once the cancellation succeeds, remove the volume from the drive.

3.5.3.3 MVTIMEOUT and TAPE_MVTIMEOUT System Parameters

The MVTIMEOUT system parameter for disks and the TAPE_MVTIMEOUT system parameter for tapes define the time (in seconds) that is allowed for a pending mount verification to complete before it is automatically canceled. (See the *VMS System Generation Utility Manual* for more information on system parameters.) This parameter should always be set to a reasonable value for the typical operations at your site. Note that resetting the value of the MVTIMEOUT parameter or the TAPE_MVTIMEOUT parameter will not affect a mount verification that is currently in progress.

Ten minutes (600 seconds) is usually a good value for MVTIMEOUT and TAPE_MVTIMEOUT, whether you usually operate with or without an operator.

When a pending mount verification is canceled by timing out, OPCOM prints a message in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Mount verification aborted for device 'device-name'.
```

After a mount verification times out, all pending and future I/O requests to the volume will fail. Thus, the disk must be dismounted and remounted before it can be accessed again.

Note: Mount verification caused by a write-lock error will not time out.

4 Backing Up Files and Volumes

By duplicating files, the Backup Utility (BACKUP) protects data from loss or corruption. If a file is accidentally deleted or a disk becomes corrupted, you can use BACKUP copies to restore files or to recreate the contents of a disk volume or volume set.

This chapter describes *online* BACKUP, which runs under the control of the VMS operating system. *Standalone* BACKUP, which is bootstrapped into main memory and does not require the direction of the operating system, is described in your VAX processor installation and operations guide. See the *VMS Backup Utility Manual* for a detailed description of BACKUP and its command qualifiers, parameters, and restrictions.

Save files on a regular basis using BACKUP. This ensures that you will be able to restore files if a disk is damaged or if files are deleted accidentally. The sensitivity of the data and the frequency of modifications determine how frequently you should complete BACKUP tasks, as well as the types of BACKUP tasks you should perform.

You can perform six types of tasks using BACKUP: *copy, save, restore, compare, list,* and *journal.* To determine your BACKUP needs, you should understand when and why to perform each type of BACKUP task. Copy, save, and restore operations are mutually exclusive. That is, BACKUP can perform only one of these tasks at a time. You can create a journal file during a save operation by specifying the command qualifier /JOURNAL. You can combine a compare operation with a save, restore, copy, or list operation by specifying the command qualifier /VERIFY. You can combine a list operation with any other BACKUP task by specifying the command qualifier /LIST.

The result of a BACKUP command depends on the type of media used. You can create copies in standard VMS file format on a Files–11 structured disk, or you can create BACKUP *save sets* on magnetic tape, a Files–11 structured disk, or a sequential disk. You can restore a save set from a magnetic tape, a Files–11 structured disk, or a sequential disk to standard VMS file format on a Files–11 structured disk.

Enter BACKUP commands at the DCL command level. For most BACKUP tasks, the command line includes an *input specifier* and an *output specifier.* The input and output specifiers identify the input to and output from BACKUP. In addition, you can specify qualifiers in the command line to modify BACKUP's default behavior.

4.1 An Overview of BACKUP Tasks

You can perform six types of BACKUP tasks: copy, save, restore, compare, list, and journal. The input and output specifiers in the BACKUP command line, along with any qualifiers specified, determine which task BACKUP performs. Table 4–1 briefly describes each BACKUP task. Section 4.7 contains detailed explanations and examples of each BACKUP task.

Table 4–1 BACKUP Tasks

Task	Explanation
Save	BACKUP save operations safeguard data against accidental deletion or disk corruption. Save operations create save sets and place the contents of selected disk files, directories, volumes, or volume sets into the save set. A save set is a file created by BACKUP and written in a format that only BACKUP can interpret. (A save set stored on a Files–11 disk is a standard VMS file, however, and can be copied, renamed, deleted, or backed up.)
Restore	BACKUP restore operations return data saved during a BACKUP save operation to its original VMS file format. You can either restore all files and directories to a disk volume or volume set, or you can restore a subset of files and directories to a disk volume or volume set.
Copy	BACKUP copy operations create equivalent copies of files, directories, volumes, or volume sets on local Files–11 disks. You cannot make BACKUP copies on magnetic tapes, Files–11 disks attached to a remote node, or sequential disks.
Compare	BACKUP compare operations verify the integrity of a file or volume created by BACKUP. Perform a compare operation to check the integrity of a file or volume created in a BACKUP save, copy, or restore operation.
List	BACKUP list operations list the date and time a save set was created, the user name of the person who created it, and the names of the files in the save set. Because BACKUP save sets are written in a unique format that only BACKUP can interpret, the list operation is the only way to determine the contents of a save set without restoring it.
Journal	BACKUP journal operations create and list journal files, on-disk records containing the specifications of files saved during a BACKUP save operation.

4.2 An Overview of BACKUP Modes

To select and process data for a BACKUP task, BACKUP provides five operating modes. Table 4–2 briefly describes each BACKUP mode. Section 4.7 contains detailed explanations and examples of BACKUP tasks performed in each mode.

Table 4–2 BACKUP Modes

BACKUP Mode	Explanation
File	Copies, saves, restores, lists, or compares files, directories, or directory trees.
Selective	Copies, saves, restores, lists, or compares files or volumes selectively, according to criteria such as version number, file type, UIC, date and time of creation, expiration date, or modification date. You can use wildcard characters and input file-selection qualifiers to perform selective BACKUP operations.
Incremental	An incremental save operation saves files that were created or modified since the last save operation. An incremental restore operation restores files from an incremental save set to a disk.
Image	An image save operation, also called a full backup, creates a functionally equivalent copy of an entire volume or volume set. An image restore operation initializes the output disk and restores an entire volume or volume set. An image copy operation initializes the output disk and copies an entire volume or volume set. An image compare operation compares entire volumes or volume sets. Although an image operation requires more time to perform than a physical operation, it offers the following two advantages: • When an image operation to a disk is performed, all files on the output disk are stored contiguously. Contiguous storage of files eliminates disk fragmentation and creates contiguous free blocks of disk space. • You can restore files and directories selectively from an image save set.
Physical	Copies, saves, restores, or compares an entire volume. However, instead of creating a functionally equivalent duplicate, it creates an exact duplicate. That is, it ignores the file structure on the disk and processes by logical disk blocks. A save set that is created with the command qualifier /PHYSICAL must also be restored using the /PHYSICAL qualifier.

4.3 Understanding the BACKUP Command Line

Use the DCL command BACKUP to perform BACKUP operations. A BACKUP command usually includes an input parameter and an output parameter, as follows:

```
BACKUP input-specifier output-specifier
```

BACKUP evaluates the input and output specifiers to determine which type of operation it is to perform.

You can use several types of qualifiers to modify the default behavior of BACKUP. You can modify the action of the entire command, or you can change the way BACKUP processes the input and output specifiers. *Command qualifiers* modify the command itself, *input-specifier qualifiers* modify the processing of the input specifier, and *output-specifier qualifiers* modify the processing of the output specifier.

There are two types of input-specifier qualifiers: *input file-selection qualifiers* and *input save-set qualifiers.* Input file-selection qualifiers allow you to select specific files from the input specifier in a save, copy, compare, or list operation. Input save-set qualifiers change the way BACKUP processes an input save set during a restore operation. There are also two types of output-specifier qualifiers: *output file qualifiers* and *output save-set qualifiers.* Output file qualifiers affect the way BACKUP copies or restores files to a Files–11 structured disk volume. Output save-set qualifiers affect the processing of an output save set during a save operation.

The position of qualifiers in the BACKUP command line affects the results of the command. Although command qualifiers can be placed anywhere in the command line, input- and output-specifier qualifiers are position-dependent. That is, input-specifier qualifiers must be placed immediately after the input specifier, and output-specifier qualifiers must be placed immediately after the output specifier. Additionally, several BACKUP qualifiers can be used in more than one way. Therefore, to achieve the desired results from a BACKUP command, take care when using position-dependent qualifiers.

For example, the /SAVE_SET qualifier can be used as an output save-set qualifier in a BACKUP save operation, and as an input save-set qualifier in a BACKUP restore operation.

4.4 Using BACKUP Media

You can perform BACKUP tasks using magnetic tapes or disks. BACKUP can process files in standard VMS format on Files–11 disks and save sets on magnetic tapes, Files–11 disks, and sequential disks.

4.4.1 Using BACKUP with Magnetic Tape

Magnetic tape is the most commonly used medium for storing BACKUP save sets. It is less expensive than disk media, and its compact size makes it easy to store. You can use more than one tape device at a time to save or restore data; this allows processing to continue on another tape while the tape used most recently is rewinding.

BACKUP treats all magnetic tape files as BACKUP save sets. You cannot use save-set specifications as both the input and output specifiers in a BACKUP command line. Therefore, you cannot specify a magnetic tape in both the input and output specifier.

Save-set specifications on magnetic tape are limited to 17 characters, including the period delimiter (.) and file type. Do not specify a directory or a version number in a magnetic tape save-set specification. The following is a valid save-set specification:

```
WKLY23SEPDLY.BCK
```

4.4.1.1 Automatic Tape Unloading

When you attempt a BACKUP save operation to a write-protected magnetic tape, BACKUP automatically unloads the tape and displays a WRITENABLE message. After the tape unloads, you can remove it from the drive, insert a write ring, and replace the tape in the drive. BACKUP displays the WRITENABLE message on your terminal if you specified the command qualifier /NOASSIST or on the operator terminal if you did not specify /NOASSIST. From the operator terminal, enter the REPLY/TO command to restart the save operation. From your terminal, enter YES to restart the save operation.

4.4.1.2 Tape Label Processing

By default, BACKUP processes information stored in the volume header record of the tape before writing to a magnetic tape. The volume header record contains volume protection information, an expiration date, and a volume label. By processing the volume protection information, BACKUP ensures that you have the right to access the volume in the manner you requested. By processing the tape expiration date, BACKUP prevents you from initializing a magnetic tape that has not yet expired. By comparing the volume label specified in the BACKUP command line to the volume label of the tape, BACKUP prevents you from creating a save set on the wrong magnetic tape.

You can prevent BACKUP from processing the tape expiration date and the volume label by specifying the command qualifier /IGNORE=LABEL_PROCESSING.

4.4.1.3 Initializing Magnetic Tapes

You must initialize a new magnetic tape to prepare it to receive data and to write a volume label, tape expiration date, and volume protection data to the volume header record. You can initialize a used magnetic tape to remove access to data stored on the tape, change the volume label, change the tape expiration date, change the volume protection data on the tape, and prepare the tape to receive new data. If a magnetic tape contains a non-ANSI or non-ISO label, initialize the tape to write an ANSI label to the volume header record.

You can use either the DCL command INITIALIZE or the BACKUP output save-set qualifier /REWIND to initialize a magnetic tape. To initialize a volume that was previously initialized with the output save-set qualifiers /REWIND and /PROTECTION, you must either own the volume (your UIC matches the owner UIC of the volume), or you must have VOLPRO privilege.

Initializing Magnetic Tapes with the DCL Command INITIALIZE

The DCL command INITIALIZE first searches the volume header record for a tape expiration date and verifies that the tape is expired. If the tape is expired, the INITIALIZE command writes a new volume label to the tape. By writing a new volume label to the tape, BACKUP initializes the tape, removing access to any data that previously resided on the tape and preparing the tape to receive new data. If the tape is not expired, the following error message appears:

```
%INIT-F-FILNOTEXP, file is not expired
```

Use the DCL command INITIALIZE/OVERRIDE=EXPIRATION to initialize a magnetic tape that has not yet expired. You need either VOLPRO privilege or write access to the volume, or you must be the owner of the volume to use the /OVERRIDE=EXPIRATION qualifier.

Initializing Magnetic Tapes with the /REWIND Qualifier

The output save-set qualifier /REWIND causes BACKUP to rewind to the beginning of the magnetic tape and check the volume protection of the tape. If you do not have access to the tape, BACKUP displays the following error message:

```
%BACKUP-F-OPENOUT, error opening 'tape:[000000]save-set-name'
-SYSTEM-F-NOPRIV, no privilege for attempted operation
```

If you have access to the tape, BACKUP searches the volume header record for a volume label and compares the label specified in the BACKUP command with the volume label. If the volume header record contains no volume label, BACKUP writes the label specified in the BACKUP command to the volume header record, initializing the tape. BACKUP then creates the save set on the tape.

If no label is specified explicitly in the command line, BACKUP uses the first six characters of the save-set name as the volume label of the first tape in a multivolume save set, and the first four characters of the save-set name followed by the volume number of the tape as the volume label of subsequent tapes. To specify a label or list of labels explicitly, use the /LABEL qualifier. If you do not specify enough labels with the /LABEL qualifier, BACKUP uses the first four characters of the final label in the list, followed by the volume number of the tape, as the volume label of subsequent tapes.

If BACKUP finds a volume label on the tape, it compares the volume label with the label you specified in the BACKUP command line (either explicitly with the /LABEL qualifier or implicitly through the save set name) and ensures that the tape is expired. If the volume label is less than six characters long, BACKUP pads the volume label with the blank character to six characters. The first four characters of the volume label must either match the first four characters of the label specified in the BACKUP command line exactly, or the first four characters of the volume label must end with one or more underscore characters. If the first four characters of the volume label end with one or more underscore characters, and the label specified in the command line matches the part of the volume label that appears before the underscore characters, BACKUP accepts the match. (For example, the volume label ABN_ matches the label ABN but does not match the label ABNE.) If either the fifth or sixth character of the volume label is a number between zero and nine, BACKUP does not compare these characters with corresponding characters in the label specified in the BACKUP command line. Otherwise, the fifth and sixth characters in the volume label must match the corresponding characters in the label specified in the BACKUP command line exactly. The following table illustrates volume labels that match labels specified in the BACKUP command line:

Label Specified in the Command Line	Matching Volume Labels
MAR	MAR, MAR_, MAR_nn
MAR_	MAR_, MAR_nn
MARK	MARK, MARKnn
MARKER	MARKER, MARKnn

If the label in the BACKUP command line matches the volume label of the tape and the tape is expired, BACKUP overwrites the volume label of the tape with the same volume label. By overwriting the tape's volume label, BACKUP initializes the tape, removing access to any data that previously resided on the tape and preparing the tape to receive new data. During the initialization process, BACKUP writes the values specified with the output save-set qualifiers /TAPE_EXPIRATION, /PROTECTION, and /OWNER_UIC to the volume header record. (If these qualifiers are not specified, the default tape expiration date is today, the default protection is none, and the owner UIC of the tape is the UIC of the current process.) After initializing the tape, BACKUP writes the save set to the tape.

If the label in the BACKUP command line did not match the volume label of the tape, BACKUP displays the following message and prompt on your terminal if you specified the command qualifier /NOASSIST or on the operator terminal if you did not specify /NOASSIST:

```
%BACKUP-W-MOUNTERR, volume 'number' on 'device' was not mounted because
  its label does not match the one requested
Specify option (QUIT, NEW tape or OVERWRITE tape)
BACKUP>
```

If you enter QUIT at the BACKUP> prompt, BACKUP aborts, unloads the magnetic tape, and issues the following message:

```
%BACKUP-F-ABORT, operator requested abort on fatal error
```

If you enter NEW at the BACKUP> prompt, BACKUP unloads the magnetic tape and issues the following prompt for a new tape:

```
%BACKUP-I-READYWRITE, mount volume 'volume-number' on _'device-name': for writing
Enter "YES" when ready:
```

If you enter OVERWRITE at the BACKUP> prompt, and the tape is expired, BACKUP overwrites the old volume label with the new volume label. (OVERWRITE instructs BACKUP to ignore the fact that either the tape has not expired or that the labels do not match.) By overwriting the tape's volume label, BACKUP initializes the tape, removing access to any data that previously resided on the tape and preparing the tape to receive new data. During the initialization process, BACKUP writes the values specified with the output save-set qualifiers /TAPE_EXPIRATION, /PROTECTION, and /OWNER_UIC to the volume header record. After initializing the tape, BACKUP writes the save set to the tape.

If the tape is not expired, BACKUP displays the following message and prompt on your terminal if you specified the command qualifier /NOASSIST or on the operator terminal if you did not specify /NOASSIST:

```
%BACKUP-W-MOUNTERR, volume 'number' on 'device' was not mounted because
  its expiration date is in the future
Specify option (QUIT, NEW tape or OVERWRITE tape)
BACKUP>
```

Specify QUIT to abort the BACKUP operation and unload the magnetic tape. Specify NEW to direct BACKUP to prompt for a new tape. Specify OVERWRITE to ignore the fact that the tape has not expired and to initialize the tape. After initializing the tape, BACKUP writes the save set to the tape.

4.4.1.4 Protecting a Magnetic Tape Volume

By default, BACKUP applies no protection to magnetic tapes. You can assign a UIC protection code to a magnetic tape by specifying the /PROTECTION qualifier with the /REWIND qualifier. All save sets that you create on the tape will be protected with this protection code. If you specify no code with the /PROTECTION qualifier, BACKUP assigns the default protection of the current process to the tape. The following command specifies a protection code that allows full access to SYSTEM and OWNER and no access to GROUP and WORLD for the save set FRIDAY.DAT and all savesets created subsequently on the magnetic tape labeled DLY101:

```
$ BACKUP
_FROM: []
_TO: MTA0:FRIDAY.BCK/REWIND/LABEL=DLY101-
_$ /OWNER_UIC=[301,211]/PROTECTION=(S:RWED,O:RWED,G:,W:)
```

As shown in the preceding example, you can also use the output save-set qualifier /OWNER_UIC to assign a UIC value to the output tape. If you do not specify the /OWNER_UIC qualifier, the output tape receives the UIC value of the process that created the save set.

4.4.1.5 Tape Expiration Dates

You can specify an expiration date for a magnetic tape volume by using the output save-set qualifiers /REWIND and /TAPE_EXPIRATION. For example, your daily incremental BACKUP tapes should expire in seven days, and your weekly incremental BACKUP tapes should expire in one month. If you do not specify /TAPE_EXPIRATION, today's date is used. The following example assigns an expiration date of January 1, 1989 to the output tape:

```
$ BACKUP
_FROM: [MILADY]
_TO: MTA0:ANNUAL.BCK/REWIND/LABEL=DLY101-
_$ /TAPE_EXPIRATION=01-JAN-1989
```

4.4.1.6 Assigning Volume Labels to Magnetic Tapes

Magnetic tape volume labels can contain a maximum of six characters. You can use any ANSI "a" character in a magnetic tape volume label. The ANSI "a" characters include numbers, uppercase letters, and any one of the following nonalphanumeric characters:

```
! " % ' ( ) * + , _ . / : ; < = > ?
```

If you use any of the preceding nonalphanumeric characters, you must enclose the volume label with quotation marks.

Label your magnetic tapes according to the data contained on the tapes. The following table presents some suggestions for labeling tapes.

Label	Type of Backup	Expiration Date
DLY101	Daily, group 1, volume number 1	Expires in 7 days
DLY102	Daily, group 1, volume number 2	Expires in 7 days
WKY101	Weekly, group 1, volume number 1	Expires in 4 weeks
WKY201	Weekly, group 2, volume number 1	Expires in 4 weeks
MTH101	Monthly, group 1, volume number 1	Expires in 12 months
YRY101	Yearly, group 1, volume number 1	Expires in 5 years

4.4.1.7 Automatic Tape Mounting

BACKUP requires a magnetic tape to be mounted as a foreign volume before it can perform operations using a tape. BACKUP automatically issues mount requests for the tapes it needs. Therefore, you do not need to use the DCL command MOUNT to mount magnetic tapes before performing BACKUP operations. After mounting a tape, BACKUP compares the label specified in the BACKUP command to the volume label in the volume header record of the tape.

If no label is specified explicitly in the command line, BACKUP uses the first six characters of the save-set name as the volume label of the first tape in a multivolume save set, and the first four characters of the save-set name followed by the volume number of the tape as the volume label of subsequent tapes. To specify a label or list of labels explicitly, use the /LABEL qualifier. If you do not specify enough labels with the /LABEL qualifier, BACKUP uses the first four characters of the final label in the list followed by the volume number of the tape as the volume label of subsequent tapes.

BACKUP compares the volume label with the label you specified in the BACKUP command line (either explicitly with the /LABEL qualifier or implicitly through the save-set name) and ensures that the tape is expired. If the volume label is less than six characters long, BACKUP pads the volume label with the blank character to six characters. The first four characters of the volume label must either match the first four characters of the label specified in the BACKUP command line exactly, or the first four characters of the volume label must end with one or more underscore characters. If the first four characters of the volume label end with one or more underscore characters, and the label specified in the command line matches the part of the volume label that appears before the underscore characters, BACKUP accepts the match. (For example, the volume label ABN_ matches the label ABN but does not match the label ABNE.) If either the fifth or sixth character of the volume label is a number between zero and nine, BACKUP does not compare these characters with corresponding characters in the label specified in the BACKUP command line. Otherwise, the fifth and sixth characters in the volume label must match the corresponding characters in the label specified in the BACKUP command line exactly. If the labels match, BACKUP performs the designated operation.

If the volume label specified in the BACKUP command line does not match the volume label of the tape, BACKUP displays the following message and prompt:

```
%BACKUP-W-MOUNTERR, volume 'number' on 'device' was not mounted because
  its label does not match the one requested
Specify option (QUIT, NEW tape or OVERWRITE tape)
BACKUP>
```

Specify QUIT to abort the BACKUP operation and unload the magnetic tape. Specify NEW to direct BACKUP to prompt for a new tape. Specify OVERWRITE to ignore the label mismatch and write the save set to the tape.

By default, BACKUP applies the command qualifier /ASSIST and displays the BACKUP-W-MOUNTERR message and the associated prompt on operator terminals enabled to receive TAPES and CENTRAL messages. When using BACKUP interactively, you can specify /NOASSIST to prevent operator terminals from receiving the message and prompt; the message and prompt appear on your terminal. (The /[NO]ASSIST qualifier is the same as the Mount Utility (MOUNT) qualifier /[NO]ASSIST.) In batch mode, BACKUP ignores the /NOASSIST qualifier and sends mount failure messages to operator terminals.

You can also mount magnetic tape volumes explicitly before beginning a BACKUP operation to tape. This ensures that BACKUP command procedures that were written before automatic tape mounting was implemented execute correctly.

4.4.1.8 Using the /NOREWIND Qualifier

By default, BACKUP uses the output save-set qualifier /NOREWIND when you create a save set on a magnetic tape. The /NOREWIND qualifier directs BACKUP to create the save set at the logical end-of-tape, rather than rewinding and initializing the tape before creating the save set. BACKUP compares the label specified in the BACKUP command line with the volume label in the volume header record of the magnetic tape before creating the save set. (If the current tape position is not the beginning-of-tape, BACKUP reads the volume label of the tape from an operating system data structure rather than rewinding to the beginning-of-tape to check the volume label.)

If no label is specified explicitly in the command line, BACKUP uses the first six characters of the save-set name as the volume label of the first tape in a multivolume save set, and the first four characters of the save-set name followed by the volume number of the tape as the volume label of subsequent tapes. To specify a label or list of labels, explicitly use the /LABEL qualifier. If you do not specify enough labels with the /LABEL qualifier, BACKUP uses the first four characters of the final label in the list followed by the volume number of the tape as the volume label of subsequent tapes.

BACKUP compares the volume label with the label you specified in the BACKUP command line (either explicitly with the /LABEL qualifier or implicitly through the save-set name) and ensures that the tape is expired. If the volume label is less than six characters long, BACKUP pads the volume label with the blank character to six characters. The first four characters of the volume label must either match the first four characters of the label specified in the BACKUP command line exactly, or the first four characters of the volume label must end with one or more underscore characters. If the first four characters of the volume label end with one or more underscore characters, and the label specified in the command line matches the part of the volume label that appears before the underscore characters, BACKUP accepts the match. (For example, the volume label ABN_ matches the label ABN but does not match the label ABNE.) If either the fifth or sixth character of the volume label is a number between zero and nine, BACKUP does not compare these characters with corresponding characters in the label specified in the BACKUP command line. Otherwise, the fifth and sixth characters in the volume label must match the corresponding characters in the label specified in the BACKUP command line exactly. If the labels match, BACKUP winds the tape forward to the logical end-of-tape (the end of the last save set stored on the tape) and creates the save set on the tape.

If the labels do not match, you can direct BACKUP to write the save set to the tape by selecting the OVERWRITE option at the BACKUP> prompt.

You can specify the output save-set qualifier /REWIND to direct BACKUP to initialize the tape. Section 4.4.1.3 describes the output save-set qualifier /REWIND.

You can use the input save-set qualifier /REWIND to rewind the tape before restoring the save set from a magnetic tape.

4.4.2 Using BACKUP with Disks

You can perform BACKUP operations with local Files–11 disks, Files–11 disks attached to a remote node, and sequential disks. The following sections describe how to use BACKUP with disks.

4.4.2.1 Using BACKUP with Local Files–11 Disks

You can use BACKUP to copy files, directories, or directory trees from one local Files–11 disk to another, or from one directory tree to another on the same Files–11 disk. BACKUP preserves directory structures when it copies directory trees. To copy files on disks, enter a BACKUP command without specifying the output save-set qualifier /SAVE_SET.

You can create save sets on a local Files–11 disk using standard Files–11 format. This type of save set is called a *Files–11 save set.* If you are writing to a disk volume set, all input and output volumes in the set must be mounted before you begin the save operation. When writing Files–11 save sets to disk, use the output save-set qualifier /SAVE_SET in the BACKUP command line. If you do not use the output save-set qualifier /SAVE_SET, BACKUP copies the specified files to standard Files–11 format files instead of creating a Files–11 save set.

BACKUP can read a Files–11 save set as either a Files–11 save set or a sequential-disk save set. When BACKUP reads a save set as a Files–11 save set, all volumes of the save set must be mounted. To read a save set that is not located in your process default directory, you must specify the directory in which the save set is located. When BACKUP reads a Files–11 save set as a sequential-disk save set, you can mount the disks containing the save set one volume at a time using the /FOREIGN qualifier to the MOUNT command. You must also specify the master file directory [000000] in the save-set specification when reading a Files–11 save set as a sequential-disk save set.

4.4.2.2 Using BACKUP with Sequential Disks

Sequential disks are local Files–11 disks that are mounted as foreign volumes and that contain only save sets. Sequential disks allow you to use Files–11 disks sequentially, the way magnetic tapes are used; you can mount multivolume sequential-disk save sets one volume at a time. This is particularly useful in configurations that have no tape drive but have a large fixed-media disk and a small removable disk. When one sequential disk is full, BACKUP prompts you to mount another disk. You can use more than one disk device at a time to save or restore data; this allows processing to continue on another disk while the one most recently used is spinning down.

Before creating or restoring a sequential-disk save set, mount the first volume using the DCL command MOUNT/FOREIGN. Although the disk is mounted with the /FOREIGN qualifier, BACKUP manages the disk using Files–11 structure.

When you perform a save operation to a sequential disk, you must use the output save-set qualifier /SAVE_SET. If you do not specify /SAVE_SET, BACKUP displays the following error message:

```
%BACKUP-F-MOUNTFLL, 'device:[000000]filename' must be mounted Files-11
```

4.4.2.3 Using BACKUP with Files–11 Disks Attached to a Remote Node

You can write or read a *network save set* on a Files–11 disk attached to a remote node by specifying the node name in the save-set specification. A remote node is a node that is accessible over a network to the node you are working on (the host node). The network save set must be located on a publicly accessible disk (a disk mounted from the remote node with the /SYSTEM, /GROUP, or /CLUSTER qualifier) on the remote node. Depending on the volume and file protection at the remote node, you may need to specify an access control string in the network save-set specification. An access control string includes the user name and password and has the following format:

```
remote_nodename"username password"::device_name:[directory]
```

Omit the access control string if it is not required to gain access to the remote node, such as in the case of proxy network access. See the *VMS Networking Manual* for more information about access control strings and proxy network access.

4.5 Saving Files Stored on Public Volumes

Public volumes are volumes mounted with the /SYSTEM, /GROUP, or /CLUSTER qualifiers. All system users granted access to a public volume share the volume. Because many users' files are stored on public volumes, it is important to regularly save files stored on public volumes.

Use a combination of incremental and full (image) save operations to safeguard data stored on public volumes. An incremental save operation saves only those files that have been created or modified since the last save operation. A full backup (usually an image save operation) saves all files on a volume. Periodic full backups are necessary to provide the basis for the reconstruction of a lost volume. If a volume is lost, you must first restore the most recent full backup and then restore incremental save sets performed since the last full backup. The most efficient way to restore incremental save sets is in reverse chronological order.

Perform incremental save operations more frequently than full backups. After consulting with users of the system, the system manager can decide how frequently to save files and volumes and how long to retain BACKUP media.

The following sample schedule for backing up public disk volumes provides adequate data protection for many installations:

- Daily—A daily incremental save set retained for 7 days. This schedule requires 7 daily sets of magnetic tapes that are rotated on a weekly basis.
- Weekly—A weekly incremental save set retained for 4 weeks. This schedule requires 4 weekly sets of magnetic tapes that are rotated every 4 weeks.

- Monthly—A monthly full backup (usually an image save set) retained for a year. This schedule requires 12 monthly sets of magnetic tapes that are rotated annually.

You can perform save operations either to magnetic tape or to disk. The advantage of using magnetic tape is the lower media cost, which permits you to retain save sets longer than you would if you were keeping them on disk.

However, there are several advantages to keeping full backups on disk that might outweigh the higher cost of disk media:

- Disks exhibit better data reliability.
- Disks tend to degrade more slowly in storage.
- If you have to restore data from its backup medium, an image copy stored on disk can be mounted for immediate use, whereas you must first restore a magnetic-tape save set to disk.
- If you use disks for your full backups, you can use a *rotating backup set* as described in Section 4.6.
- You can use command procedures to automate save operations to disks that do not require operation intervention.

4.6 Rotating Backup Sets

If you use disks for full backups, you can use a rotating backup set in which several disks or sets of disks are used in rotation on the system. At the end of each period of use (for example, once a month), the volume or volume set currently in use is copied to the oldest set of disks, the current volume is retired, and the new copy is put on line for use during the next period.

A rotating backup set offers the following major advantages:

- Your BACKUP copy (the volume or volume set just retired) is known to be good because it has been in use. Subsequent use of the new volume will confirm its integrity. If there are any defects in the new volume, you can copy affected files from the backup volume.
- The free space on the new volume is contiguous, and all the files on it are stored contiguously, resulting in better file system performance.

The disadvantages to a rotating backup set are as follows:

- Rotating backup sets are more vulnerable to disk errors than sets created by retiring the copy and continuing to use the original. A disk error during the copy operation may result in corrupted data on your new volume; disk errors in directories or file header records will result in the loss of one or more files. You must monitor the copy operation very carefully for errors, and manually repair any problems.
- You cannot perform a copy operation while users are updating files. The volume or volume set must be write-locked so that the copy will be consistent. This restriction only applies to rotating backup sets; it does not apply when you make a copy that will be used as a BACKUP copy. (When making a BACKUP copy, use the command qualifier /IGNORE=INTERLOCK so that BACKUP will save files that are open.

The saved files will be incomplete, but this ensures that BACKUP will save files that are always open.)

- User files created with explicit placement lose their placement when the volume is copied. This means you should not use a rotating backup set if database files that were placed for optimum performance are stored on the volume.
- Rotating backup sets are expensive to apply to fixed-media disks (as opposed to removable media).

4.7 Performing BACKUP Tasks

This section describes how to perform the BACKUP tasks commonly used in daily system operations: saving, restoring, copying, comparing files or volumes, listing the contents of save sets, and creating and listing BACKUP journal files.

4.7.1 Save Operations to Magnetic Tape

The following sections describe how to save disk files and directories to magnetic tapes. Before creating a save set on magnetic tape, BACKUP ensures that the tape is not write-locked and that the user has write access to the volume. Read Section 4.4.1 before attempting to create BACKUP save sets on magnetic tape. (Section 4.4.1 describes how to initialize magnetic tapes, and explains the automatic mounting of magnetic tapes and tape label processing performed by BACKUP.) Section 4.7.5 describes how to save entire disk volumes and volume sets to magnetic tape.

By default, BACKUP uses the output save-set qualifier /NOREWIND in a magnetic tape save operation. The /NOREWIND qualifier directs BACKUP to write the save set at the logical end-of-tape (the end of the last save set stored on the tape). If the save set continues to another magnetic tape, BACKUP initializes subsequent magnetic tapes.

If you specify the output save-set qualifier /REWIND, BACKUP rewinds to the beginning-of-tape and initializes the tape (making data that previously resided on the tape unavailable). If the save set continues to another magnetic tape, BACKUP initializes subsequent magnetic tapes. (Section 4.4.1.3 describes how BACKUP initializes tapes.)

When creating a save set on a streaming tape device, you can improve performance by increasing the number of I/O buffers used in the save operation to 5. Enter the command qualifier /BUFFER_COUNT=5 to increase the number of I/O buffers. When creating a save set on a nonstreaming tape device, leave the buffer count at its default value of 3.

Do not specify a directory or a version number in a magnetic tape save-set specification. Although the save-set specification can include any file type, the file type BCK is usually used to identify the file as a save set. Save-set specifications on magnetic tape are limited to 17 characters.

4.7.1.1 Saving Files to Magnetic Tape

You can create a save set that consists of a single disk file on a magnetic tape by entering a command with the following format:

BACKUP filename save-set-specification[/LABEL=label-name]/[NO]REWIND

The following command saves a file named CONTRACTS.DAT to a save set named CONTRACTS.BCK:

```
$ BACKUP/NOASSIST CONTRACTS.DAT MTAO:CONTRACTS.BCK/LABEL=LEGL01
```

The preceding BACKUP command determines if the volume label of the tape in MTA0 is LEGL01. If the volume label is LEGL01, BACKUP winds the tape to the logical end-of-tape and saves the disk file CONTRACTS.DAT to the save set CONTRACTS.BCK. If the volume label is not LEGL01, you can direct BACKUP to write the save set to the tape by specifying the OVERWRITE option at the BACKUP> prompt.

To save a list of files, separate the file names with commas as shown in the following example:

```
$ BACKUP/NOASSIST
_FROM: PAPERFILE.TXT,DBAO:[METALFILE]NAILFILE.DAT
_TO: MTA1:MYFILES.BCK/LABEL=FILS01/REWIND/TAPE_EXPIRATION=27-OCT-1988
```

If the volume header record has no volume label, BACKUP writes the volume label FILS01 to the volume header record and initializes the tape. Otherwise, BACKUP compares the volume label of the tape with the label FILS01, ensures that the tape is expired, and initializes the tape. (If the volume label is not FILS01 or the tape is not expired, you can change the label to FILS01 or ignore the tape expiration date by entering OVERWRITE at the BACKUP> prompt.) After the tape is initialized, BACKUP writes an expiration date of October 27, 1988 to the volume header record and saves the file PAPERFILE.TXT from your current default directory and the file DBA0:[METALFILE]NAILFILE.DAT to a save set named MYFILES.BCK.

By default, BACKUP saves all versions of each file. If you want to save only the most recent versions of files in a directory, enter a command in the following format:

BACKUP [directory]*.*; save-set-specification/LABEL=label-name/[NO]REWIND

The semicolon at the end of the file specification directs BACKUP to save only the most recent version of each file, as shown in the following example:

```
$ BACKUP/NOASSIST [AUDIOFILE...]*.*; MTAO:SOUNDS.BCK/LABEL=FILES2
```

If the volume label of the tape mounted in MTA0 is FILES2, BACKUP winds the tape to the logical end-of-tape and saves the most recent version of each file in the directory [AUDIOFILE] and its subdirectories to a save set named SOUNDS.BCK. (If the volume label is not FILES2, you can specify the OVERWRITE option at the BACKUP> prompt to direct BACKUP to write the save set to the tape.)

4.7.1.2 Saving Directories to Magnetic Tape

To save the contents of a directory to magnetic tape, either specify the directory to be saved explicitly or use empty brackets ([]) to represent the current default directory. For example, the following command creates a save set consisting of all files in the current default directory:

```
$ BACKUP/NOASSIST [] MTA0:OCT12SAVE.BCK/LABEL=ALLO01
```

If the volume label of the tape mounted in MTA0 is ALLO01, BACKUP winds the tape to the logical end-of-tape and saves all files in the current default directory to a save set named OCT12SAVE.BCK. (If the volume label is not ALLO01, you can specify the OVERWRITE option at the BACKUP> prompt to direct BACKUP to write the save set to the tape.)

4.7.1.3 Saving a Directory Tree to Magnetic Tape

To save a directory tree to magnetic tape, specify the directory tree in the input specifier. For example, the following command creates a save set that contains all files and subdirectories of the directory tree [ACCOUNTS...]:

```
$ BACKUP/NOASSIST/JOURNAL
_FROM: [ACCOUNTS...]
_TO: MTA0:OCT12ACC.BCK/LABEL=ACCT
```

If the volume label of the tape mounted in MTA0 is ACCT, BACKUP winds the tape to the logical end-of-tape and saves all files in the directory [ACCOUNTS] and its subdirectories to the save set OCT12ACC.BCK. (If the volume label is not ACCT, you can specify the OVERWRITE option at the BACKUP> prompt to direct BACKUP to write the save set to the tape.) BACKUP writes the specifications of all files saved in this save operation to the default journal file SYS$DISK:[]BACKUP.BJL.

4.7.2 Save Operations to Disks

You can write a save set to a disk in one of two formats: standard Files–11 disk structure or sequential disk structure. In either case, when writing a save set to a disk, use the output save-set qualifier /SAVE_SET. The /SAVE_SET qualifier indicates that you want BACKUP to create a save set on the output disk, rather than a copy of the file in standard VMS file format.

Enter a command in the following format to save a disk file to a disk save set:

```
BACKUP filespec ddcu:save-set-specification/SAVE_SET
```

The appropriate procedure for mounting the target disk depends on whether the save set will be written in standard Files–11 format or in sequential-disk format. If the save set will be written in standard Files–11 format, the target disk must be mounted as a Files–11 disk. If a save set will be written in sequential-disk format, the target disk must be mounted as a foreign device by specifying the command qualifier /FOREIGN to the DCL command MOUNT.

4.7.2.1 Saving to Local Files–11 Disk Volumes

To write a save set on a local Files–11 disk, first enter a command in the following format to mount the target disk:

MOUNT ddcu volume-label

If the disk is already mounted, you do not need to mount it again.

The commands in the following example mount a standard Files–11 disk in drive DRA2 and save the file CONTRACTS.DAT from your current default directory to a save set named CONTRACTS.BCK in the directory [SAVE]:

```
$ MOUNT DBA2 volume-label
$ BACKUP CONTRACTS.DAT DBA2:[SAVE]CONTRACTS.BCK/SAVE_SET
```

The output save-set qualifier /SAVE_SET is necessary for writing the output file in BACKUP save-set format. If you do not specify /SAVE_SET, BACKUP copies the input file to an output file in standard VMS file format.

4.7.2.2 Saving to Files–11 Disk Volumes Attached to a Remote Node

You can write a network save set on a Files–11 disk attached to a remote node by specifying the node name in the save-set specification. A remote node is a node that is accessible over a network to the node you are working on (the host node). The network save set must be located on a publicly accessible disk (a disk mounted from the remote node with the /SYSTEM, /GROUP, or /CLUSTER qualifier) on the remote node. Depending on the volume and file protection at the remote node, you may need to specify an access control string in the network save-set specification. An access control string includes the user name and password and appears in the file specification as follows:

remote_nodename"username password"::device_name:[directory]

The following command saves all files in the directory [BILBO] to a save set on DBA0, which is connected to the remote node DOUBLE:

```
$ BACKUP
_FROM: [BILBO]
_TO: DOUBLE"BILBO GOLDRING"::DBAO:[BILBO2]SAVEIT.BCK/SAVE_SET
```

4.7.2.3 Saving to Sequential Disk Volumes

You can save data in sequential-disk save sets. A sequential-disk save set allows you to treat disk volumes like magnetic tape volumes, mounting multivolume save sets one volume at a time. Section 4.4.2.2 describes sequential disks.

To write save sets on a sequential disk, first enter a command in the following format to mount the disk as a foreign device:

MOUNT/FOREIGN ddcu

The following commands mount and initialize a sequential disk on drive DMA0 and save the directory DBA2:[USER] to a sequential-disk save set named CONTRACTS.BCK:

```
$ MOUNT/FOREIGN DMAO:
$ BACKUP/LOG/INITIALIZE DBA2:[USER] DMAO:CONTRACTS.BCK/SAVE_SET
%BACKUP-S-COPIED, copied DBA2:[USER]CONTRACTS.DAT;1
%BACKUP-S-COPIED, copied DBA2:[USER]FORMS.DAT;1
%BACKUP-S-COPIED, copied DBA2:[USER]LAWSUITS.DAT;1
```

BACKUP writes sequentially to all output volumes mounted as foreign volumes. The command qualifier /INITIALIZE directs BACKUP to initialize the target disk, destroying all previous structure information and enabling the disk to be overwritten. The command qualifier /LOG causes BACKUP to display a list of the files on your terminal as they are saved. The output save-set qualifier /SAVE_SET instructs BACKUP to create a save set.

When you use the /INITIALIZE qualifier to initialize a disk volume, BACKUP uses the first 12 characters of the save set name as the volume label. This volume label remains in effect until you change it with the DCL command SET VOLUME/LABEL or until the volume is reinitialized.

Note: **Do not specify the command qualifier /BUFFER_COUNT when saving files to sequential disks. Specify /BUFFER_COUNT only when saving files to streaming tape devices.**

4.7.2.4 Writing Multivolume Sequential-Disk Save Sets

You can create multivolume sequential-disk save sets. This is useful if your system has no tape drive but has a large fixed-media disk and a small removable disk. Mount the first volume of the sequential-disk save set as a foreign volume before you begin the save operation.

When a sequential disk is full, BACKUP prompts you to mount another disk. You can use more than one disk device at a time to save or restore data. This allows processing to continue on another disk while the one most recently used is spinning down.

You need the user privilege LOG_IO or PHY_IO to read or write a multivolume sequential-disk save set.

The following example shows how to use more than one disk device at a time to save data to a sequential-disk save set. Note that you include the save-set specification in the first element of the output specifier list only.

```
$ MOUNT/FOREIGN DLA0:
$ BACKUP/JOURNAL  DRA2: DLA0:SAVE.BCK/SAVE_SET,DLA2:
```

The preceding command saves all data on DRA2 to a sequential-disk save set named SAVE.BCK and writes the file specifications of all files saved in this save operation to the default journal file SYS$DISK:[]BACKUP.BJL. When DLA0 is full, processing continues on disk DLA2 while DLA0 spins down. If DLA2 becomes full, you can insert a fresh disk into DLA0, and processing continues on DLA0.

BACKUP normally does not initialize the first sequential-disk volume, as the default is /NOINITIALIZE. Continuation volumes, however, are initialized. Specify the command qualifier /INITIALIZE to initialize the first volume of a sequential-disk save set. If you do not specify /INITIALIZE, the following restrictions apply to the first volume of the sequential disk save set:

- The disk must be Files–11 Structure Level 2.
- The disk must not be part of a volume set.
- The cluster factor of the disk must be 1.
- The free space on the disk cannot be fragmented into more than 100 contiguous extents.
- The index file cannot be extended.
- The master file directory cannot be extended.

Note: A set of disks written as a BACKUP sequential-disk set is referred to as a loosely coupled volume set. That is, it lacks some of the informational structures present in a normal volume set, such as the volume set list file. Because of the subtle differences in the structure, do not write files onto a sequential-disk volume as if it were a normal Files-11 disk. Obscure errors may result. Because the sequential-disk volumes are part of a volume set, they cannot be processed individually by the Analyze/Disk_Structure Utility.

4.7.3 Selective Save Operations

You can use a selective save operation to save a subset of your files to a magnetic tape or disk save set. Select files according to criteria such as version number, file type, UIC, date and time of creation, expiration date, or modification date. Use wildcard characters and input file-selection qualifiers when performing selective save operations. The input file-selection qualifiers are /BACKUP, /BEFORE, /BY_OWNER (/OWNER_UIC), /CREATED, /EXCLUDE, /EXPIRED, /MODIFIED, and /SINCE.

Note the following restriction when using wildcard characters in selective save operations. The wildcard characters denoting latest versions of files (;) and relative versions of files (;-n) are treated as the all-version wildcard specification (;*) with the input file-selection qualifier /EXCLUDE.

You can use the asterisk wildcard character (*) to select files by criteria such as file name, file type, or version number. For example, the following command selects all files with a version number of 3 and saves them in the Files-11 save set [SAVE]ALLTHREE.SAV on DMA1:

```
$ BACKUP/LOG *.*;3 DMA1:[SAVE]ALLTHREE.SAV/SAVE_SET
```

The command in the following example saves all files in the directory [SALLY.MEM] with UIC [300,16] to a Files-11 save set on DLA2:

```
$ BACKUP/LOG
_FROM: [SALLY.MEM]/BY_OWNER=[300,16]
_TO: DLA2:[USER]300_16.SAVE/SAVE_SET
%BACKUP-S-CREATED, created DLA2:[USER]ASSIGN.LIS;1
%BACKUP-S-CREATED, created DLA2:[USER]BBALL.DIS;19
%BACKUP-S-CREATED, created DLA2:[USER]BBLEAGUE.FISTATEMENT;1
%BACKUP-S-CREATED, created DLA2:[USER]BRULES.LIS;1
%BACKUP-S-CREATED, created DLA2:[USER]GYM.MAI;1
%BACKUP-S-CREATED, created DLA2:[USER]INFO.DAT;2
%BACKUP-S-CREATED, created DLA2:[USER]MEMO.MAI;1
%BACKUP-S-CREATED, created DLA2:[USER]NEWS.MAI;1
%BACKUP-S-CREATED, created DLA2:[USER]TEAM.LIS;19
```

Note that the brackets are required when you specify the UIC value. You can specify the UIC of the current default process by entering the input file-selection qualifier /BY_OWNER without a UIC value.

To select input files according to their date of creation, use the /CREATED qualifier with /BEFORE or /SINCE. The following commands save all files in the directory [TRUBBLE] that were created on or since April 3, 1988 to a sequential-disk save set [SPRS]PROBLEMS.SAV:

```
$ MOUNT/FOREIGN DBA2:
$ BACKUP
_FROM: DMA1:[TRUBBLE]*/SINCE=03-APR-1988/CREATED
_TO: DBA2:[SPRS]PROBLEMS.SAVE/SAVE_SET
```

To select files according to the expiration date recorded in the file header record, use the /EXPIRED qualifier with /BEFORE or /SINCE. The following command saves all versions of all files in the current default directory with the file type COB and an expiration date earlier than July 20, 1988 to a magnetic tape save set named JUL20SAVE.BCK:

```
$ BACKUP/NOASSIST
_FROM: []*.COB;*/BEFORE=20-JUL-1988:/EXPIRED
_TO: MFA0:JUL20SAVE.BCK/LABEL=DLY02
```

First BACKUP checks that the tape label is DLY02 and winds to the logical end-of-tape (after the end of the last save set stored on the tape). If the label is not DLY02, you can specify the OVERWRITE option at the BACKUP> prompt to direct BACKUP to write the save set to the tape. Then BACKUP creates the save set.

In some cases, you may find it practical to exclude certain files from a save operation. The following example directs BACKUP to save all files in the directory [OSCAR] and its subdirectories except for those with a file type of LOG and those with the file name NOTES. The parentheses are required when a list is specified with the /EXCLUDE qualifier.

```
$ BACKUP/NOASSIST
_FROM: DBA2:[OSCAR...]/EXCLUDE=(*.LOG;*,NOTES.*;*)
_TO: MTA1:SAVE.BCK/LABEL=WKLY01
```

4.7.4 Incremental Save Operations

An incremental save operation saves only those files that have been created or modified on a volume or volume set since the last save operation. Incremental save operations take less time to perform than operations that save all files, and require fewer magnetic tape or disk volumes to store each save set.

To perform incremental operations to save files created or modified since the last save operation, use the command qualifier /RECORD and the input file-selection qualifier /SINCE=BACKUP. The command qualifier /RECORD directs BACKUP to record the current date and time in the BACKUP date field of each file's header record. (To use the command qualifier /RECORD, you must own the files or have the user privilege SYSPRV.) The /SINCE=BACKUP qualifier directs BACKUP to select only those files that have been created or modified since the last BACKUP/RECORD operation.

The following command saves all files that have been created or modified since the last save operation to a save set named 19OCT.BCK on a magnetic tape labeled DLY01. The command qualifier /JOURNAL directs BACKUP to record the file specifications of all files saved in this save operation to the default BACKUP journal file SYS$DISK:[ ]BACKUP.BJL.

```
$ BACKUP/RECORD/JOURNAL
_FROM: DB2:[*...]/SINCE=BACKUP
_TO: MTA0:19OCT.BCK/LABEL=DLY01
```

Note: If you use the command qualifier /RECORD to perform incremental save operations on a disk volume, do not allow other users to use /RECORD in

their BACKUP operations on the same disk volume. If other users specify the /RECORD qualifier, the dates in the BACKUP date fields of file header records will change. This will make it impossible for you to save all files created or modified since you last performed a save operation.

4.7.4.1 Daily Incremental Save Operations

To perform daily incremental save operations, follow the instructions in the previous section for incremental save operations. You may also want to include the command qualifier /IGNORE=INTERLOCK. The /IGNORE=INTERLOCK qualifier instructs BACKUP to save files even if they are open for writing; this means that you may produce a copy of a file in a partial state. If you do not use /IGNORE=INTERLOCK, any files that are open at the time of the save operation are not saved. Note that you must have the user privilege SYSPRV, a system UIC, or you must own the volume to use the IGNORE=INTERLOCK qualifier.

The /IGNORE=INTERLOCK qualifier is especially useful if you have files that are always open (and would otherwise never be backed up).

The following example saves all files created or modified since the last save operation to a save set named INCD12JUN.BCK on the tape labeled DLY03:

```
$ BACKUP/IGNORE=INTERLOCK/RECORD/SINCE=BACKUP
_FROM: PUBLIC:[*...]
_TO: MTA0:INCD12JUN.BCK/LABEL=DLY03
```

4.7.4.2 Weekly Incremental Save Operations

Perform weekly incremental save operations the same way you perform daily incremental save operations. However, the input file-selection qualifier /SINCE specifies the date of the last weekly incremental save operation, normally one week earlier.

The following example assumes that the current date is October 14, 1988:

```
$ BACKUP/IGNORE=INTERLOCK/RECORD/SINCE=7-OCT-1988
_FROM: PUBLIC:[*...]
_TO: MTA0:INCW14JUN/LABEL=WKLY03
```

4.7.5 Image Save Operations

An image save operation, also called a full backup, creates a save set containing an entire volume or volume set. If the output volume is a disk, all files on the output volume are stored contiguously. Contiguous storage of files eliminates disk fragmentation and creates contiguous free blocks of disk space. This section describes how to perform image save operations to both disks and magnetic tapes.

Use the command qualifier /IMAGE to perform an image save operation. To use /IMAGE in a save operation, you either need write access to the volume index file (INDEXF.SYS) and the bit map file (BITMAP.SYS), or the input medium must be write-locked. BACKUP opens the index file to synchronize with the file system (no update is made). Finally, you must have read access to all files on the input medium.

An image save operation saves all files on the input disk, including files marked for deletion and lost files (files without a directory entry). Therefore, you cannot use input file-selection qualifiers in an image save operation. Also, the input specifier of an image save operation must be a device name.

You cannot use directory or file specifications in the input specifier of an image save operation.

During an image save operation, BACKUP creates a *save-volume summary record* on the volume to which it writes the save set. This record contains data necessary for initializing the disk volume to which the image save set is restored.

Specify the command qualifier /RECORD if you are performing the image save operation in conjunction with incremental save operations that use /RECORD. The /RECORD qualifier writes the date and time of the save operation into each file's header record. When you perform the next incremental save operation, only files created or modified since the image backup are saved.

The following example shows an image save operation from a Files–11 disk to magnetic tape:

```
$ BACKUP/IMAGE/RECORD DRA1:  MTA0:1JUN.BCK/REWIND/LABEL=DLY101
```

If you use magnetic tape as the BACKUP medium, you may need to mount additional magnetic tapes. The number of magnetic tapes depends on the size of the disk being saved and the amount of space in use on the disk. By default, BACKUP applies the /NOREWIND qualifier to a magnetic tape save operation and does not initialize the first tape in the save set. Subsequent tapes are initialized, however.

The following example shows an image save operation from a Files–11 disk to another Files–11 disk. Before you perform this save operation, you must create the directory on the target disk to which you will write the save set. (To create a top-level directory, you must have write access to the master file directory [000000]. To write to the master file directory, you must have either SYSPRV or BYPASS privilege, or the master file directory must contain an access control list entry allowing you to write to it.)

```
$ SET DEFAULT DB4:[000000]
$ CREATE/DIRECTORY [BACKUPS]
$ SET DEFAULT DMA1:[000000]
$ SHOW DEFAULT
DMA1:[000000]
$ BACKUP/IMAGE/RECORD DMA1: DB4:[BACKUPS]21JANDMA1.BCK/SAVE_SET
```

The preceding example creates a directory named [BACKUPS] on the target disk, DB4. The user sets default to the input disk, DMA1, and enters a BACKUP command to save the contents of DMA1 to the Files–11 save set named 21JANDMA1.BCK.

If your save set will fill more than one disk, you may want to create a sequential-disk save set rather than a Files–11 save set. To create a sequential-disk save set, mount the target disk with the DCL command MOUNT/FOREIGN. Then enter a BACKUP command similar to the one specified in the previous example, but do not specify a directory. BACKUP writes sequential-disk save sets to the master file directory [000000]. When the disk is full, BACKUP prompts you to mount another disk. By default, BACKUP does not initialize the first disk in a sequential-disk save set. Subsequent disks are initialized, however. (You can specify the command qualifier /INITIALIZE to direct BACKUP to initialize the first disk in a sequential-disk save set. Section 4.7.2.4 describes restrictions that apply to the first disk in a sequential-disk save set if you do not specify /INITIALIZE.)

You can save an entire volume set by following the same procedures outlined for saving a disk volume. To do so, use the device name for the root volume (volume number 1) or the logical name for the volume set as the input specifier.

4.7.6 Physical Save Operations

A physical save operation duplicates a volume by logical blocks, ignoring the file structure of the volume. Use the command qualifier /PHYSICAL to perform a physical save operation. The following command saves the contents of DMA0 to a save set named ALL_DMA0.BCK on DMA1:

```
$ BACKUP/PHYSICAL DMA0: DMA1:ALL_DMA0.BCK/SAVE_SET
```

On physical save operations between disks, the output disk must be either the same type of device as the input disk or a larger-capacity disk. Also, the output disk must not have a bad block in any location that corresponds to a good block on the input disk. (This restriction does not apply to RA-series disks.)

You can also perform physical save operations to magnetic tape. The following command saves the entire contents of DMA0 to a save set named PHYSBACK.BCK on the tape labeled MAR01:

```
$ BACKUP/PHYSICAL DMA0: MTA2:PHYSBACK.BCK/LABEL=MAR01
```

A save set created with the command qualifier /PHYSICAL must also be restored physically; that is, by including the /PHYSICAL qualifier in the command that restores it.

4.7.7 Restore Operations

A BACKUP restore operation returns data from a BACKUP save set to its original VMS file format on the specified output disk. Restore files, directories, volume, or volume sets if they have been inadvertently deleted or if the disks containing the original data have been lost or corrupted.

Enter a command in the following format to restore all files in a save set:

```
BACKUP save-set-specification[/SAVE_SET] output-specifier
```

Use the input save-set qualifier /SAVE_SET when you are restoring from a disk volume. The output specifier can be a device, directory, file name, or wildcard character.

You can also restore a subset of files from a save set. Use the input save-set qualifier /SELECT to select specific files to be restored from the input save set.

The following sections describe how to perform restore operations.

4.7.7.1 Restoring Save Sets from Magnetic Tape

You can restore all files in the save set to the directory tree from which they were saved. The following example restores all files in the save set named NOV12SAVE.BCK to the directory tree from which they were saved:

```
$ BACKUP TAPE:NOV12SAVE.BCK/REWIND [*...]
```

The /REWIND qualifier directs BACKUP to rewind the tape to the beginning-of-tape before beginning the restore operation. This ensures that the save set will be restored, even if it is located before the current tape position.

You can specify another directory tree to which the files will be restored. The following command restores all files in a magnetic-tape save set named NOV13SAVE.BCK to the directory tree [LYKINS...]:

```
$ BACKUP TAPE:NOV13SAVE.BCK/REWIND [LYKINS...]
```

You can restore specific files from a save set by using the input save-set qualifier /SELECT. In the following example, the file STRAT1.DAT in the directory [LYKINS.GLENDO] was accidentally deleted. The user had previously saved the file in the magnetic-tape save set NOV2SAVE.BCK. This example restores the file STRAT1.DAT to its original directory by using the input save-set qualifier /SELECT:

```
$ MOUNT/FOREIGN MTA0:
%MOUNT-I-MOUNTED, NOV25A mounted on _MTA0:
$ BACKUP
_From: MTA0:NOV2SAVE.BCK/SELECT=[LYKINS.GLENDO]STRAT1.DAT;5
_To:   STRAT1.DAT;5
$ DIRECTORY STRAT1.DAT
Directory [LYKINS.GLENDO]

STRAT1.DAT;5

Total of 1 file.
$
```

If you omit a save-set name for an input magnetic tape, BACKUP reads the first save set it encounters on the magnetic tape and stops processing when it reaches the end of that save set. Because BACKUP does not automatically rewind until it reaches the end-of-tape marker (EOT), you can proceed to the next save set (if there is one) by repeating the BACKUP command. You can also include the asterisk wildcard character (*) with the device specification, which directs BACKUP to read all of the save sets on the magnetic tape volume.

When restoring a save set from a streaming tape device, you can improve performance by increasing the number of I/0 buffers used in the restore operation to 5. Enter the command qualifier /BUFFER_COUNT=5 to increase the number of I/O buffers. When restoring a save set from a nonstreaming tape device, leave the buffer count at its default value of 3.

4.7.7.2 Restoring from Save Sets on Files–11 Disks

You can restore files from a Files–11 save set on a local Files–11 disk or from a Files–11 disk that is attached to a remote node. Mount the disk from which you will restore the save set unless that disk is already mounted. (Disks attached to a remote node must be mounted from the remote node with the /SYSTEM, /GROUP, or /CLUSTER qualifier.) If the save set is stored on more than one disk, all input and output disks must be mounted before you begin the restore operation. Remember to specify the input save-set qualifier /SAVE_SET. If you do not specify /SAVE_SET, BACKUP copies the save set rather than restoring its contents to standard VMS file format.

The following example restores a file named MARK.DAT (from the network save set FRIENDS.BCK in the directory [MYDIR] on the disk DB1 connected to the remote node GAMETE) to your current default directory. DB1 is already mounted.

```
$ BACKUP
_FROM: GAMETE"RONNIE THISGAME"::DB1:[MYDIR]FRIENDS.BCK/SELECT=MARK.DAT
_TO: NEWMARK.DAT
```

You can also restore Files–11 save sets as if they are sequential disk save sets. You must specify the directory from which the save set is to be read. If the save set was written to a volume set using Files–11 disk structure, the following rules apply when you restore save sets sequentially:

- All disks in the output volume set must be mounted before the restore operation begins.
- You should mount the input volume set one volume at a time, using the /FOREIGN qualifier to the MOUNT command.
- When you specify the volumes in the input specifier, specify the names of the devices in the same order as the relative volume numbers of the volumes mounted on those devices. For example, the volumes named USER01 and USER02 in a volume set are mounted on two drives, DRA3 and DRA5. Therefore, the save-set specification for the volume set is as follows:

  DRA3:[directory]save-set-name.ext;v,DRA5:

4.7.7.3 Restoring from Save Sets on Sequential Disks

You can restore a sequential-disk save set sequentially, or you can restore it as you would a save set on a Files–11 disk. This section describes how to restore a sequential-disk save set sequentially. Follow instructions in Section 4.7.7.2 to restore a sequential-disk save set as a Files–11 disk save set.

To restore the save set sequentially, mount the input volume set one volume at a time. The default directory for the save set specification is the master file directory on the disk [000000]; therefore, you do not need to include the directory in the input specification.

The following example restores the sequential-disk save set named CONTRACTS.BCK from DMA0 to a file in your current default directory named CONTRACTS.DAT:

```
$ MOUNT/FOREIGN DMAO: SAVEWORK
$ BACKUP/LOG DMAO:CONTRACTS.BCK/SAVE_SET  CONTRACTS.DAT
%BACKUP-S-CREATED, created DBA2:[USER]CONTRACTS.DAT;1
```

Note: Do not specify the command qualifier /BUFFER_COUNT when restoring files from sequential disks. Specify /BUFFER_COUNT only when restoring files from streaming tape devices.

4.7.7.4 Restoring Files to the Directory from Which They Were Saved

BACKUP does not automatically restore files to the directory from which they were saved. To restore files to the directory from which they were saved, use the directory wildcard character [*...] in the output specifier. The following example restores a magnetic tape save set named NOVELS.BCK from a Files–11 save set on DUA0 to the directories from which the files were saved on DUA1:

```
$ BACKUP DUA0:[BCKUP]NOVELS.BCK/SAVE_SET DUA1:[*...]
```

4.7.7.5 Restoring Files from Multivolume Save Sets

You might need to restore a small number of files from a multivolume save set. If your save set encompasses more than one tape or sequential-disk volume, it is usually possible to begin the restore operation by mounting the volume that contains the files, rather than the first volume of the save set. (Because an image restore operation restores the contents of the entire volume or volume set, processing must begin with the first volume if you use the command qualifier /IMAGE in your restore operation.)

If you used the command qualifier /JOURNAL when you saved the files, you can use the BACKUP/LIST/JOURNAL command to find the volume that contains the files. (Section 4.7.11 describes how to create and list journal files.) You can then mount the volume and use BACKUP to select and restore the desired files.

If the mounted tape or sequential disk is not the first volume in the save set, you receive the following warning message, which has no effect on the restore operation:

```
%BACKUP-W-NOT1STVOL, 'name' is not the start of a save set
```

4.7.7.6 Restoring Entire Disk Volumes

You can restore an entire disk volume if the disk volume was destroyed, lost, or corrupted. To restore all files from a save set, enter a command in the following format:

```
BACKUP save-set-specification[/SAVE_SET] device-specification
```

Use the input save-set qualifier /SAVE_SET when the save-set specification refers to a disk volume. The /SAVE_SET qualifier is not needed with tape volumes because BACKUP assumes the data on a magnetic tape is a save set.

4.7.7.7 Restoring a Disk Volume from an Image Save Set

An image restore operation creates a functionally equivalent copy of the volume or volume set saved in the image save operation. To restore an entire disk volume from an image save set, mount the target volume (the disk to which you will restore the save set) using the DCL command MOUNT /FOREIGN. Then use the BACKUP/IMAGE command to restore the volume. All files on the target disk will be stored contiguously. Contiguous storage of files eliminates disk fragmentation and creates contiguous free blocks of disk space.

An image restore operation initializes the target volume. By default, the target volume is initialized using the initialization parameters in the save-volume summary record on the input volume. If you specify the command qualifier /NOINITIALIZE, the target volume is initialized using volume initialization parameters found on the target volume. Use /NOINITIALIZE after changing volume initialization parameters, as described in Section 4.7.7.8.

Note: **When a volume is initialized, all access to information previously stored on the volume is lost.**

The following example restores an image save set named 24JUNDMA1.BCK on a tape to DRA1, using the parameters in the save-volume summary record to initialize DRA1:

```
$ MOUNT/FOREIGN DRA1
%MOUNT-I-MOUNTED, 24JUND mounted on _DRA1:
$ BACKUP/IMAGE MTA0:24JUNDMA1.BCK  DRA1:
```

To restore an image save set to a volume set, mount all the target volumes of the set as foreign volumes. Specify the list of devices on which the target volumes are mounted in the BACKUP command output specifier. For example:

```
$ MOUNT/FOREIGN DRA0:
%MOUNT-I-MOUNTED, mounted on _DRA0:
$ MOUNT/FOREIGN DRA1:
%MOUNT-I-MOUNTED, mounted on _DRA1:
$ MOUNT/FOREIGN DRA2:
%MOUNT-I-MOUNTED, mounted on _DRA2:
$ BACKUP/IMAGE MTA0:31JUNPUB DRA0:,DRA1:,DRA2:
```

The volumes receive volume set numbers in the order in which they are listed in the BACKUP command. In this example, DRA0, DRA1, and DRA2 become volumes 1, 2, and 3, respectively.

4.7.7.8 Changing Volume Initialization Parameters Before Restoring

Use the following procedure to change volume initialization parameters:

1. Initialize the new volume using the new initialization parameters.
2. Mount the new volume using the /FOREIGN qualifier.
3. Restore the volume with BACKUP using the command qualifiers /IMAGE and /NOINITIALIZE.

In the following example, the cluster size of the volume is changed to 3. The other volume initialization parameters take the default values from the INITIALIZE command.

```
$ ALLOCATE DRA2
%DCL-I-ALLOC, _DRA2: allocated
$ INITIALIZE/CLUSTER_SIZE=3 DRA2: TEST_PROGS
$ MOUNT/FOREIGN DRA2
%MOUNT-I-MOUNTED, TEST_PROGS mounted on _DRA2:
$ BACKUP/IMAGE/NOINITIALIZE/TRUNCATE  MTA0:1JUN.BCK  DRA2:
```

If you reduce the cluster size initialization parameter, you may choose to specify the command qualifier /TRUNCATE in the BACKUP command that restores files to the volume, as shown in the preceding example. When you reduce the cluster size of a volume, each file consumes a number of blocks equal to an even multiple of the old cluster size. To reduce the wasted space to a minimum, the command qualifier /TRUNCATE causes BACKUP to truncate the restored file at the end-of-file (EOF). Using /TRUNCATE prevents BACKUP from allocating empty clusters of disk blocks to files when the file allocation exceeds the actual file size. (The cluster size is the minimum number of blocks that can be allocated to a file. If the cluster size for the disk is greater than one, a file may use more space than is required because the cluster size determines how many blocks are allocated to a file.)

The only initialization parameter that you cannot change is the Files–11 structure level of the volume.

4.7.7.9 Restoring a Volume from Incremental Backups

If you have been performing a combination of full and incremental save operations on a volume, use the following procedure to recover the volume if it has been lost, corrupted, or destroyed.

First, restore the volume from the last image save set using an image restore operation. The following example restores a multivolume magnetic tape save set named FULLJUN84.BCK to DRA0. The command qualifiers /RECORD and /IMAGE are required for the correct operation of this procedure.

```
$ MOUNT/FOREIGN DRA0:
%MOUNT-I-MOUNTED, mounted on _DRA0:
$ BACKUP/IMAGE/RECORD MTA0:FULLJUN84.BCK,MTA1 DRA0:
%BACKUP-I-RESUME, resuming operation on volume 2
%BACKUP-I-RESUME, resuming operation on volume 3
%BACKUP-I-RESUME, resuming operation on volume 4
      .
      .
      .
$ DISMOUNT/NOUNLOAD DRA0:
```

Next, mount the disk as a file-structured volume and restore all incremental save sets created since the image save set was created. The most efficient way to restore incremental saves sets is in reverse chronological order. Start with the last daily incremental save set. Then restore the preceding daily incremental save sets, and finally, the weekly incremental save sets, as follows:

```
$ MOUNT DRA0: PUBLIC
%MOUNT-I-MOUNTED, PUBLIC mounted on _DRA0:
! Mount the tape containing the save set INCD17JUN in drive MTA0
$ BACKUP/INCREMENTAL  MTA0:INCD17JUN DRA0:
! Remove the tape containing the save set INCD17JUN from drive MTA0
! Mount the tape containing the save set INCD16JUN in drive MTA0
$ BACKUP/INCREMENTAL  MTA0:INCD16JUN DRA0:
! Remove the tape containing the save set INCD16JUN from drive MTA0
! Mount the tape containing the save set INCD15JUN in drive MTA0
$ BACKUP/INCREMENTAL  MTA0:INCD15JUN DRA0:
! Remove the tape containing the save set INCD15JUN from drive MTA0
! Mount the tape containing the save set INCW14JUN in drive MTA0
$ BACKUP/INCREMENTAL  MTA0:INCW14JUN DRA0:
! Remove the tape containing the save set INCW14JUN from drive MTA0
! Mount the tape containing the save set INCW7JUN in drive MTA0
$ BACKUP/INCREMENTAL  MTA0:INCW7JUN DRA0:
```

If you choose to selectively exclude certain files in your incremental save operations (for example, listing files or batch logs), these files will not be restored but will have directory entries in the resulting volume. You can delete these null directory entries by running a repair pass with the Analyze/Disk_Structure Utility. (See the *VMS Analyze/Disk_Structure Utility Manual*.)

If directory files were renamed during the time period covered by the incremental save sets, these directories appear on the reconstructed volume under both their old and new directory names. The files that were written since the directory was renamed appear under the new directory name; the other files, written before the directory was renamed, appear under the old directory name. You must merge the old and new directories manually.

If you renamed many directory files, your disk may become full during the course of the incremental restore operations. Merge the old and new directories and complete the incremental restore operations.

4.7.7.10 Restoring a Volume from a Physical Backup

A physical restore operation restores a physical save set, which is a save set created with the command qualifier /PHYSICAL. A physical restore operation restores data to the volume by logical blocks, ignoring the file structure of the volume. Use the command qualifier /PHYSICAL to perform a physical restore operation. The following command restores the contents of a physical save set named ALL_DMA0.BCK on DMA1 to DMA0:

```
$ BACKUP/PHYSICAL DMA1:ALL_DMA0.BCK/SAVE_SET DMA0:
```

On physical restore operations between disks, the output disk must be the same type of device as the disk from which the save set was created. Also, the output disk must not have a bad block in any location that corresponds to a good block on the disk from which the save set was created. (This restriction does not apply to RA-series disks.)

You can also perform physical restore operations from magnetic tape. The following command restores the entire contents of DMA0 from a physical save set named PHYSBACK.BCK on a magnetic tape:

```
$ BACKUP/PHYSICAL MTA2:PHYSBACK.BCK DMA0:
```

4.7.8 Copy Operations

A BACKUP copy operation creates equivalent copies of files, directories, volumes, or volume sets on local Files–11 disks. You cannot make BACKUP copies on remote Files–11 disks, sequential disks, or magnetic tapes.

Copies created with the BACKUP command are different from copies made with the DCL command COPY. The DCL command COPY produces new copies of files, updating the revision dates and assigning protection codes from the applicable defaults. Copies created with BACKUP are identical to the originals, including revision dates and protection codes (although, by default, the owner UIC of the copies is the UIC of the current process). Also, the BACKUP copy operation can copy entire directory trees, maintaining the directory structure, while the DCL command COPY cannot.

4.7.8.1 Copying a File

Create a BACKUP copy of a file by entering a command in the following format:

BACKUP full_file_spec full_file_spec

The full file specification can include a disk name, directory, file name, file type, and version number. BACKUP assumes the default disk and directory and uses wildcard values for the file name, file type, and version number if you do not specify the full file specification.

For example, enter the following command to make a BACKUP copy of your EDTINI.EDT file, rename it OLDEDTINI.EDT, and move it to a directory named [HARRIS.MISC]:

```
$ BACKUP/LOG EDTINI.EDT [HARRIS.MISC]OLDEDTINI.EDT
%BACKUP-S-CREATED, created DISK$DEFAULT:[HARRIS.MISC]OLDEDTINI.EDT;4
```

The command qualifier /LOG directs BACKUP to display information about the processed file. Note that the copied file retains the version number of the original file. If you do not include a version number with the input file, BACKUP processes all existing versions of the file and applies their version numbers to the respective output files. In the preceding example, only one EDTINI.EDT file exists, and it has a version number of 4.

4.7.8.2 Copying Multiple Files

You can make BACKUP copies of multiple files using a single command line. The output specification must be a directory or wildcard character; it cannot be a list of file names or a single file name. For example:

```
$ BACKUP/LOG ELEMENT.DAT,MATERIAL.DAT,SUBSTANCE.DAT [HARRIS.STUFF]
%BACKUP-S-CREATED, created DISK$DEFAULT:[HARRIS.STUFF]ELEMENT.DAT;1
%BACKUP-S-CREATED, created DISKS$DEFAULT:[HARRIS.STUFF]MATERIAL.DAT;1
%BACKUP-S-CREATED, created DISK$DEFAULT:[HARRIS.STUFF]SUBSTANCE.DAT;1
```

All files designated in the input specifier are copied to the output directory. If the output specification had attempted to rename the files by giving three new file names, the operation would have failed, and BACKUP would have returned an error message. The operation would also have failed if the output specifier was a single file name.

4.7.8.3 Selective Copy Operations

You can use a selective copy operation to copy a subset of your files. Select files according to criteria such as version number, file type, UIC, date and time of creation, expiration date, or modification date. Use wildcard characters and the following input file-selection qualifiers when performing selective copy operations: /BACKUP, /BEFORE, /BY_OWNER (/OWNER_UIC), /CREATED, /EXCLUDE, /EXPIRED, /MODIFIED, or /SINCE.

The command in the following example copies files that were created before 21-SEP-1988 with a file type of .MEM from the directory [SMITH] on DMA0 to the directory [JONES] on DMA1:

```
$ BACKUP DMA0:[SMITH]*.MEM/CREATED/BEFORE=21-SEP-1988 DMA1:[JONES]
```

4.7.8.4 Copying an Entire Directory Tree

You can copy an entire directory tree by entering a command in the following format:

BACKUP disk:[directory...] disk:[directory...]

If you do not specify a disk, BACKUP copies the file from one directory to another on your current default disk.

If the specified output directory does not exist, BACKUP creates it. To create a top-level directory, you must have write access to the master file directory [000000]. To write to the master file directory, you must have either SYSPRV or BYPASS privilege, or the master file directory must contain an access control list entry allowing you to write to it.

In the following example, BACKUP creates a directory named [OLD.STUFF] as well as subdirectories that correspond to the subdirectories of [NEW.STUFF]. BACKUP copies all files from the directory [NEW.STUFF] and its subdirectories to [OLD.STUFF] and its subdirectories.

```
$ BACKUP/LOG [NEW.STUFF...] [OLD.STUFF...]
%BACKUP-S-CREDIR, created directory DISK$DEFAULT:[OLD.STUFF]
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF]ELEMENT.LIS;1
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF]ESSENCE.LIS;1
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF]MATERIAL.LIS;1
%BACKUP-S-CREDIR, created directory DISK$DEFAULT:[OLD.STUFF.MORESTUFF]
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF.MORESTUFF]MATTER.DAT;1
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF.MORESTUFF]NOMATTER.DAT;1
%BACKUP-S-CREDIR, created directory DISK$DEFAULT:[OLD.STUFF.MORESTUFF.WHAT]
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF.MORESTUFF.WHAT]GOO.YUK;2
%BACKUP-S-CREDIR, created directory DISK$DEFAULT:[OLD.STUFF.NOMORESTUFF]
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF.NOMORESTUFF]NUTHIN.LIS;1
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF.NOMORESTUFF]ZILCH.DAT;1
%BACKUP-S-CREDIR, created directory DISK$DEFAULT:[OLD.STUFF.NOMORESTUFF.REALLY]
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF.NOMORESTUFF.REALLY]ZIP.ZAP;1
%BACKUP-S-CREATED, created DISK$DEFAULT:[OLD.STUFF]SUBSTANCE.LIS;1
```

4.7.8.5 Image Copy Operations

Use an image copy operation to copy an entire volume or volume set. Because an image copy operation processes all files on the volume or volume set, you cannot use input file-selection qualifiers in an image copy operation. Also, the input specifier of an image copy operation must be a device name. You cannot use directory or file specifications in the input specifier of an image copy operation.

Use the command qualifier /IMAGE to perform an image copy operation. When you use /IMAGE in a copy operation, by default the output volume is initialized using initialization data from the input volume. You can specify the command qualifier /NOINITIALIZE to direct BACKUP to initialize the output volume using initialization data from the target volume. All files on the target volume are stored contiguously. Contiguous storage of files eliminates disk fragmentation and creates contiguous free blocks of disk space.

To use the command qualifier /IMAGE in a copy operation, you must have either write access to both the index file (INDEXF.SYS) and the bit map file (BITMAP.SYS) on the input disk, or you must write-lock the input disk before beginning the copy operation.

4.7.8.5.1 Copying a Disk Volume

To copy an entire disk volume to another disk volume (called the target disk), first mount the target disk as a foreign volume. Enter a BACKUP command, specifying the command qualifier /IMAGE. Use only device names for the disks. Before copying the contents of the input disk, BACKUP initializes the target disk.

The following example initializes DBA2 using initialization data from DBA1 and copies all files on DBA1 to DBA2, creating a functionally equivalent copy of the entire disk. All files on DBA2 are stored contiguously.

```
$ MOUNT/FOREIGN DBA2:
$ BACKUP/IMAGE DBA1: DBA2:
```

The following procedure describes how to use BACKUP to copy all files on a public disk to another disk. (A public disk is a disk mounted with the /SYSTEM, /GROUP, or CLUSTER qualifier.)

Before you start, write-lock the source disk to prevent users from changing any data on the disk during the copy operation. Users will be able to read files on the source disk during the copy operation.

1 Enter the following command to warn all users that the disk will be dismounted so it can be write-locked for a copy operation:

```
$ REPLY/ALL/BELL "DISK will be write-locked in 10 mins. for BACKUP"
```

This message should indicate the name of the source disk and the time the write-lock will be done.

2 Use the SHOW DEVICE/FILE command to ensure that all files on the disk have been closed. Then send a message informing users with open files to close them.

3 At the time indicated in your message, enter the DISMOUNT /NOUNLOAD command to logically dismount the source disk, as follows:

```
$ DISMOUNT/NOUNLOAD input-disk:
```

4 Mount the source disk again, as follows:

```
$ MOUNT/SYSTEM/NOWRITE input-disk: volume-label
```

The /NOWRITE qualifier write-locks the source disk.

5 Place the target disk in the drive and ready the device by pressing the RUN/STOP button or the START/STOP switch.

6 Mount the target disk, as follows:

```
$ MOUNT/FOREIGN target-disk:
```

7 Enter the following BACKUP command to copy the input disk to the target disk and to compare the BACKUP copy to the original files after the copy operation completes:

```
$ BACKUP/IMAGE/VERIFY input-disk:  target-disk:
```

Section 4.7.9 contains more information about the /VERIFY qualifier.

8 At completion, dismount the target disk with the following command:

```
$ DISMOUNT target-disk:
```

9 Remove the target disk from the drive and place a label on the outside of the disk; the label should include the volume label and current date.

10 Dismount the source disk by entering the DISMOUNT/NOUNLOAD command as follows:

```
$ DISMOUNT/NOUNLOAD input-disk:
```

11 Remount the source disk with the MOUNT command, as follows:

```
$ MOUNT/SYSTEM input-disk: volume-label
```

12 Inform all users that the source disk is no longer write-locked by entering the following command:

```
$ REPLY/ALL/BELL "'Input-disk' is no longer write-locked."
```

The following example illustrates how to copy all files on a public disk to another disk:

```
$ REPLY/ALL/BELL "DMA2: WILL BE WRITE-LOCKED IN 5 MINS. FOR BACKUP."
Reply received on NODE1 from SYSTEM at NODE1_OPA0: 06:31:29
DMA2:  WILL BE WRITE-LOCKED IN 5 MINS. FOR BACKUP.
$ SHOW DEVICE/FILES DMA2:
   .
   .
   .
$ DISMOUNT/NOUNLOAD DMA2:
$ MOUNT/SYSTEM/NOWRITE DMA2: PUBLIC
%MOUNT-I-WRITELOCK, volume is write-locked
%MOUNT-I-MOUNTED, PUBLIC     mounted on _DMA2:
$ MOUNT/FOREIGN DMA1:
%MOUNT-I-MOUNTED, PUBLIC     mounted on _DMA1:
$ BACKUP/IMAGE/VERIFY DMA2: DMA1:
$ DISMOUNT DMA1:
$ DISMOUNT/NOUNLOAD DMA2:
$ MOUNT/SYSTEM DMA2: PUBLIC
%MOUNT-I-MOUNTED, PUBLIC    mounted on _DMA2:
$ REPLY/ALL/BELL "DMA2: IS NO LONGER WRITE-LOCKED."
Reply received on NODE1 from SYSTEM at NODE1_OPA0: 06:46:44
DMA2: IS NO LONGER WRITE-LOCKED.
```

4.7.8.5.2 Copying a Disk Volume Set

A disk volume set is a group of disks that shares a common directory structure. The first volume in the set is called the root volume. Each volume in the set is identified by a volume number relative to the root volume.

Before copying a disk volume set, mount all target volumes with the /FOREIGN qualifier. Use the logical name of the volume set as the input specifier; the output specifier must include a device specification for each target volume in the set. The following example creates an image copy of a two-disk volume set named WORK$. Before copying the contents of each input volume, BACKUP initializes the target volume using the initialization data from the input volume.

```
$ MOUNT/FOREIGN DBA1:,DBA2:
$ BACKUP/IMAGE WORK$ DBA1:,DBA2:
```

4.7.8.5.3 Copying a Disk Volume Set When Drives are Limited

Normally, when you use BACKUP to copy a volume set, you must mount all volumes of the input and output volume sets. Therefore, it may not be possible to copy a large volume set directly to another set of disks if you have a limited number of disk drives. The following example shows how to copy a volume set one volume at a time with the BACKUP/IMAGE /VOLUME command, using only one drive to create the output volume set. The example copies a three-volume set, so at least four drives are required. You must write-lock the input volume set during the entire procedure to ensure consistency.

```
$ REPLY/ALL/BELL "DISK$PUBLIC VOLUME SET WILL BE -
_$ WRITE-LOCKED IN 5 MINS. FOR BACKUP." ❶
Reply received on NODE1 from SYSTEM at NODE1_OPA0: 06:31:29
DISK$PUBLIC VOLUME SET WILL BE WRITE-LOCKED IN 5 MINS. FOR BACKUP.
$ DISMOUNT/NOUNLOAD DRA0: ❷
$ MOUNT/SYSTEM/NOWRITE DRA0:,DRA1:,DRA2: PUBLIC01,PUBLIC02,PUBLIC03 DISK$PUBLIC ❸
%MOUNT-I-MOUNTED, PUBLIC01 mounted on _DRA0:
%MOUNT-I-MOUNTED, PUBLIC02 mounted on _DRA1:
%MOUNT-I-MOUNTED, PUBLIC03 mounted on _DRA2:
$ MOUNT/FOREIGN DRA3: ❹
%MOUNT-I-MOUNTED, WKLY01 mounted on _DRA3:
$ BACKUP/IMAGE/INITIALIZE/VOLUME=1 DISK$PUBLIC DRA3: ❺
$ DISMOUNT DRA3: ❻
$ MOUNT/FOREIGN DRA3:
%MOUNT-I-MOUNTED, WKLY02 mounted on _DRA3:
$ BACKUP/IMAGE/INITIALIZE/VOLUME=2 DISK$PUBLIC DRA3:
$ DISMOUNT DRA3:
$ MOUNT/FOREIGN DRA3:
%MOUNT-I-MOUNTED, WKLY3 mounted on _DRA3:
$ BACKUP/IMAGE/INITIALIZE/VOLUME=3 DISK$PUBLIC DRA3:
$ DISMOUNT DRA3: ❼
$ DISMOUNT/NOUNLOAD DISK$PUBLIC ❽
$ MOUNT/SYSTEM DRA0:,DRA1:,DRA2: PUBLIC01,PUBLIC02,PUBLIC03 DISK$PUBLIC ❾
%MOUNT-I-MOUNTED, PUBLIC01 mounted on _DRA0:
%MOUNT-I-MOUNTED, PUBLIC02 mounted on _DRA1:
%MOUNT-I-MOUNTED, PUBLIC03 mounted on _DRA2:
$ REPLY/ALL/BELL "DISK$PUBLIC VOLUME SET IS NO LONGER WRITE-LOCKED." ❿
Reply received on NODE1 from SYSTEM at NODE1_OPA0: 06:46:44
DISK$PUBLIC VOLUME SET IS NO LONGER WRITE-LOCKED.
```

❶ This command warns users that DISK$PUBLIC (a disk volume set on drives DRA0, DRA1, and DRA2) is about to be write-locked for BACKUPS.

❷ To disable writing on the volume set, dismount (but do not unload) DRA0, the root volume in the volume set DISK$PUBLIC.

❸ Mount the volumes PUBLIC01, PUBLIC02, and PUBLIC03 with the /SYSTEM qualifier and specify write-locking with the /NOWRITE qualifier. In this example, the volume set is also given the logical name DISK$PUBLIC.

❹ Mount the first target volume on drive DRA3 as a foreign volume.

❺ Copy the first volume, PUBLIC01, to the target disk mounted in DRA3. BACKUP initializes the target disk, using initialization data from the input disk, before beginning the copy operation.

❻ When the volume on DRA3 is full, dismount it and mount a new target volume. Enter a BACKUP command to copy the next volume of the volume set to the target disk in DRA3. Repeat this procedure until the third volume has been copied.

❼ Dismount the last volume on DRA3.

❽ To enable writing on the volume set again, dismount (but do not unload) DISK$PUBLIC.

❾ Mount DISK$PUBLIC again with the /SYSTEM qualifier, omitting the /NOWRITE qualifier. The volume set is now available for writing.

❿ Tell users that the volume set DISK$PUBLIC is available for writing.

Note: **Do not write-lock a mounted volume; doing so is likely to cause errors or suspension of system activity. You must dismount the public volume set, remount it write-locked, and continue as described previously. (Dismounting a disk flushes data out of all file system caches and VMS RMS caches.)**

4.7.8.6 Physical Copy Operations

A physical copy operation duplicates a volume by logical blocks, ignoring the file structure of the volume. Use the command qualifier /PHYSICAL to perform a physical copy operation. The following command copies the contents of DMA0 to DMA1:

```
$ BACKUP/PHYSICAL DMA0: DMA1:
```

On all physical copy operations, the output disk must be the same type of device as the input disk; for example, a physical copy operation cannot be performed between an RP05 input disk and an RP06 output disk. Also, the output disk must not have a bad block in any location that corresponds to a good block on the input disk. (This restriction does not apply to RA-series disks.)

4.7.9 Comparing Files Using BACKUP

A BACKUP compare operation compares a save set with disk files or compares disk files with other disk files. Perform a compare operation to check the integrity of a file or volume after a copy, save, or restore operation. For example, you can compare a save set to the original files to verify that the files were saved correctly.

There are two ways to perform a compare operation. You can either perform a compare operation in conjunction with a save, restore, copy, or list operation by specifying the command qualifier /VERIFY or you can perform a compare operation independently of other BACKUP operations by specifying the command qualifier /COMPARE. When you specify /VERIFY, BACKUP first performs the save, restore, copy, or list operation and then compares the output with the input. When you use /VERIFY in a copy or list operation, BACKUP displays no message when it begins the compare operation. When you use /VERIFY in a save or restore operation, BACKUP displays the following informational message when it begins the compare operation:

```
%BACKUP-I-STARTVERIFY, starting verification pass
```

If a file does not compare successfully, BACKUP displays the following error message:

```
%BACKUP-E-VERIFYERR, verification error for block 'block-number'
  of 'disk:[directory]file_name.file_type;version_number'
```

The compare operation should be used only to compare the following:

- A save set to original files
- Files or volumes copied using BACKUP to original files

Because BACKUP processes files by blocks, comparing files not produced by BACKUP is likely to cause mismatch errors in files that are apparently identical.

The following example combines a compare operation with a copy operation:

```
$ BACKUP/VERIFY/LOG FRED.DAT [FRIENDS]OLDFRED.DAT
%BACKUP-S-CREATED, created DISK$:[FRIENDS]OLDFRED.DAT;3
%BACKUP-S-COMPARED, compared DISK$:[FRIENDS]OLDFRED.DAT;3
```

This compare operation found no inconsistencies.

You can compare a save set with files on a disk, as shown in the following example:

```
$ BACKUP/COMPARE TAPE:2MAR1555.BCK [LYKINS]
```

The preceding command directs BACKUP to compare the contents of the save set 2MAR1555.BCK with the directory [LYKINS]. You can also compare files on a disk with other files on a disk. For example:

```
$ BACKUP/COMPARE FIX.EXE;3 [SAVED]OLDFIX.EXE;3
%BACKUP-E-VERIFYERR, verification error for block 16 of SYSD$:[WORK]FIX.EXE;4
```

In this example, there is an inconsistency in block 16 between the compared files.

To compare two entire Files–11 volumes, use an image compare operation as follows:

```
$ BACKUP/IMAGE/COMPARE DBA1: DBA2:
```

To compare a physical save set with files on a disk, use a physical compare operation, as follows. All disks in a physical compare operation must be mounted as foreign volumes.

```
$ MOUNT/FOREIGN DBA2:
$ BACKUP/PHYSICAL/COMPARE MTA0:PHYSBACK.BCK DBA2:
```

4.7.10 Listing the Contents of a BACKUP Save Set

Because BACKUP save sets are written in a format that only BACKUP can interpret, the list operation is the only way to determine the contents of a save set without restoring the save set. You can either display the list on a terminal or write it to a specified output file. You can perform a list operation in conjunction with any other BACKUP operation. Use the command qualifier /LIST to perform a list operation.

By default, a list operation supplies the same information about files in the save set as the DCL command DIRECTORY/DATE/SIZE, including the actual number of blocks saved for each file. Specify the command qualifier /FULL with the /LIST qualifier to list information about files supplied by the DCL command DIRECTORY/FULL, including the number of blocks allocated for each file.

You can also perform a BACKUP list operation to list the contents of a BACKUP journal file. BACKUP journal files, which are created during save operations by using the command qualifier /JOURNAL[=file-spec], contain on-disk records of BACKUP save operations and the file specifications of the files saved during each operation. Section 4.7.11 contains more information about BACKUP journal files.

Before you can list the contents of a save set, the media containing the save set must be inserted into an appropriate drive. If the save set is stored on a disk, the disk must be mounted as a Files–11 volume or as a foreign volume. BACKUP mounts magnetic tapes automatically as part of the list operation.

The following command displays save-set information on your terminal about a magnetic tape save set. Before entering this command, the magnetic tape containing the save set was inserted into drive MTA0.

```
$ BACKUP/LIST MTA0:2MAR1555.BCK/SAVE_SET
Listing of save set(s)

Save set:           2MAR1555.BCK
Written by:         POLYANNA
UIC:                [000200,000207]
Date:               21-AUG-1988 09:36:14.68
Command:            BACKUP/LOG [USER.SAVE] MTA0:2MAR555.BCK/REWIND/LABEL=WK102
Operating system:   VMS version 5.0
BACKUP version:     5.0
CPU ID register:    08000000
Node name:          _SUZI::
Written on:         _MTA0:
Block size:         8192
Group size:         10
Buffer count:       3

[USER.SAVE]ANOTHER.DAT;1                                       1  18-AUG-1988 14:10
[USER.SAVE]LAST.DAT;1                                          1  18-AUG-1988 14:11
[USER.SAVE]THAT.DAT;1                                          7  18-AUG-1988 14:10
[USER.SAVE]THIS.DAT;2                                          1  18-AUG-1988 13:44

Total of 4 files, 10 blocks
End of save set
```

The following command lists information about a disk save set to an output file named INFO.LIS. Before this command was entered, the disk containing the save set was mounted in drive DLA1.

```
$ BACKUP/LIST=INFO.LIS DLA1:2MAR1999.BCK/SAVE_SET
```

The following command combines a list operation with a save operation to magnetic tape:

```
$ BACKUP/LIST=MYBACK.DAT [PRAMS] MTA0:2MAR1555.BCK/LABEL=DLY201
```

BACKUP verifies that the tape label is DLY201 and copies the contents of the directory [PRAMS] to a save set named 2MAR1555.BCK. The command qualifier /LIST causes BACKUP to write save-set information to the file MYBACK.DAT as the save operation proceeds.

If you do not know the names of save sets stored on a magnetic tape volume, you can use only the device specification of the drive in which the tape is inserted as the output specifier in the BACKUP/LIST command. If you only use the device specification, BACKUP reads the next save set it encounters on the magnetic tape and then stops processing when it reaches the end of that save set. BACKUP does not automatically rewind to the beginning-of-tape marker (BOT) unless you include the input save-set qualifier /REWIND in your command. Therefore, you can proceed to the next save set (if one exists) by repeating the command.

By including the asterisk wildcard character (*) with the device specification and the /REWIND qualifier, you can direct BACKUP to rewind to the beginning-of-tape and list all save sets on the tape volume, as follows:

```
$ BACKUP/LIST MTA0:*/REWIND
```

To list only the names of the save sets on a magnetic tape volume, mount the magnetic tape volume as a Files–11 volume. Then enter the DCL command DIRECTORY, as follows:

```
$ MOUNT MTA0: SAVE01 TAPE
%MOUNT-I-MOUNTED, SAVE01 mounted on _MTA0:
$ DIRECTORY/SIZE/DATE/PROTECTION TAPE:
Directory MTA0:[]

CONTRACTS.BCK;1          5  13-JUN-1988 12:11  (RWED,RWED,RE,)
JUL20SAVE.BCK;1          3  20-JUL-1988 23:59  (RWED,RWED,RE,)
MYPHILE.BCK;1            2  28-SEP-1988 12:00  (RWED,RWED,RE,)

Total of 3 files, 10 blocks.
```

You can use the DIRECTORY command to list all save sets stored on a disk if you have the READALL privilege. All files stored on the sequential disks are save sets. To list all save sets stored on a sequential disk, mount the disk as a Files–11 disk, set default to the master file directory, and enter the DIRECTORY command, as follows:

```
$ MOUNT DMA0: WKL101
$ SET DEFAULT [000000]
$ DIRECTORY
```

A Files–11 disk can contain both save sets and standard VMS files. You can identify the save sets by their file type as long as you used the same file type for all save sets. Mount the disk if it is not already mounted, set default to the master file directory, and enter the DIRECTORY command to list the save sets on the Files–11 disk. The following example lists all save sets with the file type BCK on a disk labeled PUBLIC:

```
$ MOUNT DMA1: PUBLIC
$ SET DEFAULT [000000]
$ DIRECTORY [...]*.BCK;*
```

4.7.11 Creating and Listing BACKUP Journal Files

A BACKUP journal file is an on-disk record of BACKUP save operations and the file specifications of the files saved during each operation. Use the command qualifier /JOURNAL[=file-spec] in a BACKUP save operation to create a journal file. The primary advantage of creating BACKUP journal files is that you can list information about save operations without mounting the media that contains the save sets.

If you do not include a file specification with the command qualifier /JOURNAL, the name of the BACKUP journal file defaults to SYS$DISK:[ ]BACKUP.BJL. If you include a file specification and the specified BACKUP journal file does not exist, it is created. If it already exists, the new journal information is appended to the existing journal file. You can start a new version of a BACKUP journal file (with the same file name as an existing journal file) by creating an empty file using a text editor or the DCL command CREATE.

To list the contents of a BACKUP journal file, enter a command in the following format:

BACKUP/LIST[=file-spec]/JOURNAL[=file-spec]

You cannot specify an input or output specifier with a BACKUP/JOURNAL /LIST command. If the file specification is omitted from the command qualifier /LIST, output is directed to your terminal; if the file specification is omitted from the command qualifier /JOURNAL, the default BACKUP journal file (SYS$DISK:[ ]BACKUP.BJL) is used.

You can use file selection qualifiers with the BACKUP/JOURNAL/LIST command. This allows you to locate a file or set of files in a save set. Before restoring a file or small set of files from a multivolume save set, use the BACKUP/JOURNAL/LIST command to find the volume that contains the files. You can then mount the volume and selectively restore the desired files.

The following example displays all files in the directory [SMITH.PROGS] that were saved after October 5, 1988 and listed in the BACKUP journal file ARCH.BJL:

```
$ BACKUP/LIST/JOURNAL=ARCH.BJL/SELECT=[SMITH.PROGS]/SINCE=5-OCT-1988
Listing of BACKUP journal
Journal file _DB1:[SYSMGR]BACKUP.BJL;1 on 10-OCT-1988 00:45:06.36
Save set ARCH.BJL created on 10-OCT-1988 00:45:06.36
Volume number 1, volume label WKL101

               [SMITH.PROGS]RUNTHIS.FOR;4
               [SMITH.PROGS]TIMER.PAS;5
               [SMITH.PROGS]REMINDER.FOR;46
                             .
                             .
                             .
```

The following example shows how to create a BACKUP journal file and how to list the contents of the BACKUP journal file:

```
$ BACKUP/JOURNAL/LOG/IMAGE  DRA2: MTA0:3OCT.FUL/LABEL=DLY101
%BACKUP-S-COPIED, copied DRA2:[COLLINS]ALPHA.DAT;4
%BACKUP-S-COPIED, copied DRA2:[COLLINS]EDTINI.EDT;5
   .
   .
   .
%BACKUP-I-RESUME, resuming operation on volume 2
%BACKUP-I-READYWRITE, mount volume 2 on _MTA0: for writing
Press return when ready: [RET]
%BACKUP-S-COPIED, copied DRA2:[LANE]MAIL.MAI;1
%BACKUP-S-COPIED, copied DRA2:[LANE]MEMO.RNO;5
   .
   .
   .
$ BACKUP/JOURNAL/LIST
Listing of BACKUP journal
Journal file _DB2:[SYSMGR]BACKUP.BJL;1 on 3-OCT-1988 00:40:56.36
Save set 3OCT.FUL created on 3-OCT-1988 00:40:56.36
Volume number 1, volume label DLY101

         [COLLINS]ALPHA.DAT;4
         [COLLINS]EDTINI.EDT;5
         [COLLINS]LOGIN.COM;46
         [COLLINS]LOGIN.COM;45
         [COLLINS]MAIL.MAI;1
         [COLLINS]MAR.DIR;1
         [COLLINS.MAR]GETJPI.EXE;9
         [COLLINS.MAR]GETJPI.LIS;14
                .
                .
                .
         [LANE]LES.MAI;1
                .
                .
                .
Save set 3OCT.FUL created on 3-OCT-1988 00:40:56.36
Volume number 2, volume label DLY102

          [LANE]MAIL.MAI;1
          [LANE]MEMO.RNO;5
          [LANE]MEMO.RNO;4
                .
                .
                .
          [WALTERS.VI]KD.RNO;52

End of BACKUP journal
```

4.8 Protecting a BACKUP Save Set

Limiting access to BACKUP save sets is an important part of system security. The file system treats a BACKUP save set as a single file, whether it is stored on disk or on magnetic tape. Therefore, anyone who has access to a save set can read any file in the save set. BACKUP does not check protection on individual files until after they are restored to standard VMS file format.

To maintain system security, it is crucial that you protect save sets adequately. Accordingly, you should assign restrictive protection to save sets on disk and to magnetic tape volumes by using the output save-set qualifiers /OWNER_UIC and /PROTECTION. Sufficient protection can prevent nonprivileged users from mounting a save-set volume or reading files from a save set. You should also take physical security precautions with save sets stored offline by keeping BACKUP media in locked cabinets.

When you write a save set to a Files–11 disk or a sequential disk and do not specify /PROTECTION, BACKUP applies the process default protection to the save set. If you specify /PROTECTION, any protection categories that you do not specify default to your default process protection.

Protection information is written to the volume header record of a magnetic tape and applies to all saves sets stored on the tape. Therefore, the output save-set qualifiers /OWNER_UIC and /PROTECTION are effective on magnetic tape save sets only if you specify the output save-set qualifier /REWIND. This rewinds the tape to its beginning, writes the protection data to the volume header record, and initializes the tape. If you specify /PROTECTION, any protection categories that you do not specify default to your default process protection. If you do not specify /REWIND with the /PROTECTION and /OWNER_UIC qualifiers, the magnetic tape receives no protection. (By default, BACKUP applies no protection to magnetic tapes.)

The following example saves the directory [PAYROLL] to KNOX.BCK on the magnetic tape drive MFA2. The output save-set qualifier /LABEL provides the label BANK01 for the tape. The output save-set qualifier /OWNER_UIC assigns an owner UIC of [003,003] to the save set. The output save-set qualifier /TAPE_EXPIRATION assigns an expiration date of January 15, 1989 to the tape. The output save-set qualifier /PROTECTION assigns the owner of the volume read, write, execute, and delete access. SYSTEM users are assigned read, write, and execute access; GROUP users are assigned read and execute access; WORLD users are assigned no access.

```
$ BACKUP
_FROM: [PAYROLL]
_TO: MFA2:KNOX.BCK/LABEL=BANK01/REWIND/OWNER_UIC=[003,003]-
_$ /TAPE_EXPIRATION=15-JAN-1989/PROTECTION=(S:RWE,O:RWED,G:RE,W)
```

When a nonprivileged user wants to restore a particular file, do not lend the volume containing the save set. You could give away access to all the files on the volume. The safest way to restore a particular file is to restore the file selectively, as shown in the following example:

```
$ BACKUP MTA0:JULY.BCK/SELECT=[JONES.TEXTPROC]LASTMONTH.DAT -
_$ [*...]/BY_OWNER=ORIGINAL
```

The selected file is restored with its original directory, ownership, and protection. In this way, the file system determines if the user is permitted access to the file.

4.9 Using Command Procedures to Perform Backup Tasks

If you perform similar BACKUP tasks regularly, you can write command procedures to assist you in performing these tasks. For example, if you perform daily incremental save operations to magnetic tape, you use identical or similar command strings. You could use a command procedure to enter the necessary commands, as in the following example:

```
$ ! Command procedure DAILYBACK.COM
$ !
$ ! Execute this command procedure interactively,
$ !  by entering the command @[directory]DAILYBACK
$ !  at the DCL prompt.
$ !
$ ! The BACKUP command in this procedure contains the
$ !  output save-set qualifier /REWIND.  Therefore, this
$ !  command procedure always initializes the output tape.
$ !
$ ON CONTROL_Y THEN GOTO EXIT
$ ON ERROR THEN GOTO EXIT
$ INQUIRE DRIVE "Enter the drive name (without a colon)"
$ ALLOCATE 'DRIVE'
$ INQUIRE SAVESET_SPEC "Enter the save-set specification"
$ INQUIRE LBL "Enter the tape label"
$ INQUIRE EXP "Enter the tape expiration date"
$ BACKUP/NOASSIST/RECORD/IGNORE=INTERLOCK/SINCE=BACKUP -
  [...] 'DRIVE':'SAVESET_SPEC'/REWIND/LABEL='LBL'/TAPE_EXPIRATION=''EXP'
$ EXIT:
$ DEALLOCATE 'DRIVE'
$ EXIT
```

The preceding command procedure allocates the specified tape drive. BACKUP searches the tape's volume header record for a volume label and compares the label you specified with the volume label. If the volume header record contains no volume label, BACKUP writes the label and expiration date you specified to the volume header record and initializes the tape. Otherwise, BACKUP compares the tape's volume label with the label you specified and ensures that the tape is expired. If the tape is not expired or the label does not match, the command procedure exits. If the tape is expired and the label matches, BACKUP writes the expiration date you specified to the volume header record and initializes the tape. After initializing the tape, BACKUP saves all files in the current default directory tree that have been created or modified since the last save operation to a save set with the name you specified.

Note that if you do not have an ON ERROR statement in your command procedure and BACKUP returns a FATAL or ERROR status code, the command procedure exits immediately after the BACKUP command is executed. If you want your command procedure to complete, include an ON ERROR statement as shown in the preceding procedure.

DIGITAL provides two template command procedures in the SYS$EXAMPLES directory to assist system managers in designing BACKUP command procedures. These command procedures are called BACKUSER.COM and RESTUSER.COM.

5 Managing Disk Space

The disk space that is available for files is a finite amount. As system manager, you share a responsibility with your users of making the best use of the available disk space. The following techniques are some of the methods that can be used to conserve and monitor disk space usage:

- Purge old versions of files
- Establish disk quotas
- Set file expiration dates
- Analyze and repair error conditions

The following sections discuss these tasks in detail.

5.1 Purging Old Versions of Files

One of the best ways of conserving disk space is to purge old versions of files periodically. Before purging files, make sure that the most recent versions are the ones you want to preserve.

You should periodically purge certain system log files that are automatically generated, such as the operator's log and the accounting log files. Also, purge outdated log files created by the PRINT and SUBMIT commands.

You can encourage individual users to purge files that are in their own areas and directories. If desired or necessary, the system manager can purge files from some or all directories. The following example purges all files in the directory [JONES] and all the subdirectories below [JONES] on disk $DISK1: and logs the files that are deleted (displaying their names on the terminal as they are deleted):

```
$ PURGE/LOG $DISK1:[JONES...]
```

You can use wildcard characters to perform global purges, and you can use the /KEEP qualifier to retain only a specified number of versions of each file, as shown in the following example:

```
$ PURGE/KEEP=3 $DISK1:[*...]
```

You can restrict the number of file versions that can be generated in a user's directory, by using the /VERSION_LIMIT qualifier with the SET DIRECTORY or CREATE DIRECTORY command. When you create a default directory for a user, use the /VERSION_LIMIT qualifier with the CREATE /DIRECTORY command. In the following example, files in account [JONES] cannot exceed three versions:

```
$ CREATE/DIRECTORY $DISK1:[JONES]/OWNER_UIC=[200,1]/VERSION_LIMIT=3
```

5.2 Establishing Disk Quotas

You limit the amount of space available to individual users on public volumes (or volume sets) by creating and maintaining quota files on those volumes. Individual users can similarly restrict usage on private volumes. Quotas are maintained and enforced on a per-volume basis. Each volume or volume set has its own quota file; a volume on which quotas are not maintained has no quota file; on a volume set, volume 1 contains the quota file. Each entry in a quota file includes the following information:

- General Identifier or UIC—Identification code of a user entitled to maintain files on the volume
- Usage—Number of blocks on the volume taken up by the user's files
- Quota—Maximum number of blocks on the volume that the user's files can take up before an error message is issued
- Overdraft—Number of blocks over the quota that the user's files can take up

The absolute maximum number of blocks permitted a user on a volume is the sum of the quota and the overdraft.

You (or the user maintaining the volume) supply identifiers and assign quotas and overdrafts with the Sysman Utility. (For more information, see the *VMS SYSMAN Utility Manual*). Usage counts are maintained automatically by the system during normal file activities.

The name of the quota file is [000000]QUOTA.SYS on the applicable volume. A quota file requires one block of disk storage for each 16 entries.

A quota file is initialized with an entry for UIC [0,0]. The usage count for this UIC should not change from 0—the UIC should own no files. Its quota and overdraft, however, serve as defaults in certain situations, and so should be set to values most likely to be assigned as quotas and overdrafts to other UICs.

5.2.1 Disk Quota Operations

During normal use of a volume with a quota file, the system automatically updates the usage counts as users create, delete, extend, and truncate files. Users without entries in the quota file are not allowed to create files or allocate space on the volume, unless they have the EXQUOTA privilege.

5.2.1.1 Exceeding the Quota

To create new files, a user's disk space usage must be below quota (not overdraft). If an operation to add a new file or expand a current file will exceed a user's quota, the system prohibits the operation and issues an error message.

A user with an overdraft can retry an operation involving an open file (such as an editing session). Operations to extend an open file will succeed until usage exceeds the sum of the quota and the overdraft. At this point, the system prohibits further extensions to the file.

Quota restrictions are not enforced for users with the EXQUOTA privilege; however, their usage counts are maintained.

5.2.1.2 Suspending Quotas

The DISKQUOTA DISABLE subcommand of the System Management (SYSMAN) Utility suspends quota operations on a volume; the DISKQUOTA ENABLE subcommand lifts the suspension. In addition, quota operations on a volume can be suspended at mount time by specifying the /NOQUOTA qualifier to the DCL command MOUNT. Note that when you suspend and then resume quota operations on a volume, you will probably find incorrect usage values in the quota file. To correct the usage values, use the DISKQUOTA REBUILD subcommand of SYSMAN. (See the *VMS SYSMAN Utility Manual* for more information.)

To discontinue quota operations on a volume, execute the DISKQUOTA DISABLE subcommand, exit from SYSMAN, and delete the QUOTA.SYS file. Note that quotas are not suspended across disk mounts if the QUOTA.SYS file is still present.

5.2.1.3 Ensuring Quota File Accuracy with REBUILD on Mount

When a volume is mounted that was not properly dismounted the last time it was used, the system performs an automatic REBUILD operation. If quotas are enforced on the volume, this action ensures that the quota file accurately reflects usage of the disk, under any of the following conditions:

- The system failed
- The volume was physically removed before being dismounted
- The **WRITE PROTECT** button was pushed

5.2.2 Restrictions on Other System Operations

The following restrictions and limitations apply whether or not disk quotas are being used:

- The SYSPRV privilege is required to change the owner UIC of a file, because a change in file ownership consumes the resources of another user.
- Relative volume 1 of the volume set must be on line at all times.

5.3 Setting File Expiration Dates

File expiration is a file system feature (available on Files–11 Structure Level 2 disks only) that uses the expiration date of each file to track the file's use. The expiration dates aid the disposal of seldom used files.

To enable the setting of expiration dates, use the DCL command SET VOLUME:

```
$ SET VOLUME device-name: /RETENTION=(min,max)
```

For min and max, specify the minimum and maximum retention periods for files on the volume, expressed as delta time values. For example, the following command sets the minimum retention period to 15 days and the maximum to 20 days:

```
$ SET VOLUME DRA0: /RETENTION=(15-0:0,20-0:0)
```

The retention periods operate as follows:

- Every time a user accesses a file (for either a read or write operation), and that file's expiration date is earlier than the current date plus the minimum retention period.
- The file's expiration date is updated to the current date plus the maximum retention period.

Thus, the expiration date of a frequently accessed file fluctuates between the minimum and maximum period plus the current date. When you set a suitable interval between minimum and maximum retention periods, you can trade between accuracy and efficiency in maintaining expiration dates. The difference between the two periods is the interval at which the expiration date of a frequently accessed file will be updated.

If you specify only a single value in the SET VOLUME/RETENTION command, it becomes the minimum retention period; the maximum retention period is set to twice the minimum or the minimum plus 7 days, whichever is less.

You can simulate the maintenance of "access dates," available in some other operating systems, by setting the retention periods to very small values (for example, 1 hour). Note, however, that doing so substantially increases overhead in the file system.

This feature does not automatically remove unused files; it maintains expiration dates to permit you to develop your own policy for handling files with little or no activity. For example, the following BACKUP command will copy to tape and then delete all expired files:

```
$ BACKUP/DELETE PUBLIC:[*...]/BEFORE=TODAY/EXPIRED MTAO:ARCH20JUN
```

Users may not always be aware of file expiration dates, so retain the tape for a substantial period of time.

Note: **If you start maintaining expiration dates on a previously existing volume, you should be aware that the expiration dates on existing files are zero (until the files are accessed). Files with expiration dates of zero are considered expired.**

5.4 Handling Error Conditions

This section describes two utilities that analyze errors conditions relating to disk media, ANALYZE/DISK_STRUCTURE and ANALYZE/MEDIA. You use ANALYZE/DISK_STRUCTURE to analyze disk structure errors, and you use ANALYZE/MEDIA to analyze media for bad blocks. See the specific utility manuals for detailed information.

5.4.1 Analyzing Disk Structure Errors

The DCL command ANALYZE/DISK_STRUCTURE invokes the Analyze /Disk_Structure Utility. This utility verifies the Files–11 structure on disk volumes, reports errors, and (when you specify the /REPAIR qualifier) repairs errors. By default, ANALYZE/DISK_STRUCTURE reports errors but does not make repairs. The following command reports all disk-structure errors on device DBA1:

```
$ ANALYZE/DISK_STRUCTURE DBA1:
```

You should periodically use the Analyze/Disk_Structure Utility to check for disk structure errors such as lost files on the disk. Lost files are files with a backlink to a directory file that has been deleted. (See the *VMS Analyze /Disk_Structure Utility Manual* for more information.) The command in this example analyzes and repairs all errors and lost files on device DDA0:

```
$ ANALYZE/DISK_STRUCTURE/REPAIR/CONFIRM DDAO:
```

It is a good idea to execute ANALYZE/DISK_STRUCTURE in two passes: one pass to report all errors, and a second pass with the /REPAIR and /CONFIRM qualifiers to repair selected errors.

Another opportunity to check for lost files on your system is during a Backup operation. The following BACKUP command returns a formatted listing of the files in the BACKUP save set as the operation proceeds:

```
$ BACKUP/IMAGE/LIST device_A: device_B:saveset.sav/SAVESET
```

Lost files appear at the end of the save set listing as files with empty brackets, indicating that they are not found within any directory, for example:

```
[ ]SUB_A.DIR;1
[ ]ALPHA.DAT;4
[ ]OMEGA.DAT;2

End of saveset
```

5.4.2 Analyzing Media Errors

The DCL command ANALYZE/MEDIA invokes the Bad Block Locator Utility, which analyzes block-addressable devices and records the location of blocks that cannot reliably store data.

To ensure that the device is not accessed by any other user, you must first allocate the device with the DCL command ALLOCATE. After allocating the device, enter the DCL command MOUNT/FOREIGN. This ensures that the system does not recognize the device as a Files–11 volume, therefore allowing the Bad Block Locator Utility to execute.

To test the blocks on a volume, ANALYZE/MEDIA does the following:

- Writes a test pattern to each block on the medium
- Reads the contents of the block into a buffer
- Compares the data read back with the data written

If the data does not compare exactly, a block cannot reliably store data.

When the Bad Block Locator Utility locates a bad block, it records the address of the block. Consecutive bad blocks are recorded as single entries for non-last-track devices. After it finishes testing the disk, BAD writes the addresses of the bad blocks into an area called the Detected Bad Block File (DBBF).

After the Bad Block Locator Utility locates and records the bad blocks, enter the DCL command INITIALIZE to change the volume from unstructured format to Files–11 format and to allocate the faulty blocks to a special file on the volume called [000000]BADBLK.SYS. In this way, users are protected from accessing bad blocks for their files.

Caution: **There is no way to test the volume for bad blocks without destroying its contents. However, you can update the Detected Bad Block File without erasing the volume's contents by using the ANALYZE/MEDIA qualifiers /NOEXERCISE and /BAD_BLOCKS.**

The following example shows the command sequence for executing ANALYZE/MEDIA interactively from your terminal:

```
$ ALLOCATE DBA2:
DBA2: ALLOCATED
$ MOUNT/FOREIGN DBA2:
%MOUNT-I-MOUNTED  mounted on DBA2:
$ ANALYZE/MEDIA/EXERCISE DBA2:
$ INITIALIZE DBA2:
```

The ALLOCATE command requests the allocation of a specific disk drive, DBA2. The response from the ALLOCATE command indicates that the device was successfully allocated. The MOUNT/FOREIGN command mounts the disk volume as a foreign disk. The MOUNT command response indicates that DBA2 was successfully mounted. The ANALYZE/MEDIA command invokes the Bad Block Locator Utility. Specifying DBA2 and the /EXERCISE qualifier causes the Bad Block Locator Utility to analyze each block on this disk volume and to record the bad blocks. The INITIALIZE command reformats the volume as a Files–11 volume and allocates the bad blocks to a [000000]BADBLK.SYS. Once allocated, the bad blocks cannot be used by other files.

You do not need to execute ANALYZE/MEDIA on Mass Storage Control Protocol (MSCP) devices. All MSCP devices appear to the operating system as a contiguous stream of logical blocks. If part of the disk goes bad, the drive revectors the bad block automatically. See the *VMS Bad Block Locator Utility Manual* for complete details on using the Bad Block Locator Utility.

6 Performing Batch and Print Operations

The batch and print jobs that run on a system each day often constitute a large part of the workload. As system manager, you must become familiar with the number and types of jobs that run on your system and develop ways to submit, schedule, and execute them efficiently. You manage job workloads and improve system performance by

- Setting up and managing queues
- Setting up and controlling spooled devices
- Controlling batch and print jobs

This chapter provides the concepts and procedures required to perform these tasks.

6.1 Types of Queues

The VMS batch/print facility supports two general classes of queues: execution and generic queues. Execution queues accept batch and print jobs for processing. Generic queues hold jobs until they are transferred to an execution queue for processing. Queue classes are further defined by the kind of job they accept and the type of device to which the output is directed. The following sections describe the specific types of execution and generic queues.

6.1.1 Execution Queues

An execution queue accepts jobs for processing. In a VAXcluster environment, jobs are directed to execution queues on a specific node within the cluster. The two types of execution queue, batch queue and output queue, are described in the following sections:

6.1.1.1 Batch Execution Queue

A batch execution queue can accept only batch jobs. A batch job executes as a detached process that sequentially runs one or more command procedures. You define the list of command procedures when you submit the job. Section 6.5 describes procedures specific to batch queues.

6.1.1.2 Output Execution Queue

An output execution queue accepts print jobs for processing by an independent process called a *symbiont*. The job controller sends the symbiont a list of files, which you define when you submit the job. An output symbiont transfers data from a disk to an output device. As the symbiont processes each file, it produces output for the device it controls, such as a printer or a terminal.

The standard print symbiont provided by the VMS operating system is designed to print files on hardcopy devices. User-modified or user-written symbionts can also be designed for this or any other file processing activity managed by the VMS batch/print facility. (See the *VMS Utility Routines*

Manual for more information.) Output execution queues include the following types:

- **Printer execution queue** - Uses a symbiont to direct output to a line printer.
- **Terminal execution queue** - Uses a symbiont to direct output to a terminal printer.
- **Server execution queue** - Uses a user-modified or user-written symbiont to process the files that belong to jobs in the queue.

The term **logical queue** - applies to any type of output execution queue that has had its output redirected; it is not a type of output execution queue in itself. A logical queue holds a job until it can be transferred to a specific printer, terminal, or server execution queue. Before a logical queue can transfer a job, you must first assign it to a specific output execution queue, and then start both the output execution queue and the logical queue. Section 6.6.7 describes how to set up a logical queue.

When you create an output execution queue, you can initially designate it as either a printer, terminal, or server execution queue. However, when the queue is started, the symbiont process associated with the queue can override the queue designation if the queue type specified does not match the type of device. The standard VMS symbiont determines whether it is controlling a printer or a terminal. It communicates this information to the job controller, and if necessary, the job controller changes the type designation of the output execution queue. By convention, a user-written or user-modified symbiont defines its queue as a server queue.

Section 6.6 describes procedures specific to output execution queues.

6.1.2 Generic Queues

A generic queue holds a job until an appropriate execution queue becomes available to initiate the job. The job controller then requeues the job to the available execution queue. In a VAXcluster environment, a generic queue can direct jobs to execution queues that are located on other nodes in the VAXcluster. The two types of generic queues, generic batch queue and generic output queue, are described as follows:

- **Generic batch queue -** A generic batch queue can direct jobs only to batch execution queues. Generic batch queues are typically used in clustered systems to distribute the workload across several nodes (see Section 6.5.5).
- **Generic output queue -** A generic output queue can direct jobs to any of the three types of output execution queues: print, terminal, or server. Section 6.6.8 describes how to set up a generic output queue.

6.2 Controlling the Queue Manager

The system queue manager controls the scheduling of batch and print jobs. The following sections describe how to start the queue manager and how to handle abnormal conditions.

6.2.1 Starting the Queue Manager

You start the queue manager and open the queue file with the DCL command START/QUEUE/MANAGER. A new queue file is created if one does not already exist, or if the /NEW_VERSION qualifier is specified. The START /QUEUE/MANAGER command is restricted to users with both OPER and SYSNAM privileges. You must include this command before any other queue commands in your site-specific startup command procedure, SYS$MANAGER:SYSTARTUP_V5.COM. (See the *Guide to Setting Up a VMS System* for more information on SYSTARTUP_V5.COM.)

You can specify the name of a queue file as a command parameter. If you omit this parameter, the job controller uses the default file specification SYS$SYSTEM:JBCSYSQUE.DAT. In a VAXcluster, you must include the file specification parameter to cause each system to access the same job queue manager file on a shared disk volume. The disk must be mounted before you enter the command. The following example includes a file specification parameter:

```
$ START/QUEUE/MANAGER DUA2:[SYSQUE]JBCSYSQUE.DAT
```

In this example, the queue file JBCSYSQUE.DAT is opened in the directory DUA2:[SYSQUE].

You can control the initial allocation and extension size of the queue file with the /EXTEND_QUANTITY qualifier. Specify a positive integer in the range of 10 through 65535, or 0. If 0 is specified, the default value is 100 disk blocks.

The /BUFFER_COUNT qualifier allows you to specify the number of buffers in a local buffer cache to allocate for performing I/O operations to the queue file. The default value is 50. (See the *VMS DCL Dictionary* for more information on the START/QUEUE/MANAGER command.)

6.2.2 Creating a New Queue File

In the unlikely event that the queue file (SYS$SYSTEM:JBCSYSQUE.DAT) becomes corrupted, you must create a new one. The following conditions may indicate a corrupted queue file:

- Irregularities in the display produced by the DCL command SHOW QUEUE or SHOW ENTRY
- Jobs do not process
- Jobs seem to disappear

Before you can create a new queue file, you must stop all execution queues on the system by entering the DCL command STOP/QUEUE/MANAGER. (In a cluster, you must stop the queue manager on all nodes in the cluster.) This command performs an orderly shutdown of the system job queue manager process, stops all execution queues, terminates all executing jobs, and denies any further queue requests. Once all executing batch and print jobs are terminated, the queue file is closed. The STOP/QUEUE /MANAGER command is contained in the site-independent shutdown command procedure, SYS$SYSTEM:SHUTDOWN.COM. (See the *Guide to Setting Up a VMS System* for more information on shutdown procedures.)

To create a new version of the queue file, enter the DCL command START /QUEUE/MANAGER/NEW_VERSION.

The STOP/QUEUE/MANAGER and START/QUEUE/MANAGER commands are restricted to users with both OPER and SYSNAM privileges.

6.2.3 Restarting the Queue Manager

The system job controller process is executed when the system is booted, through a command procedure called by the SYS$SYSTEM:STARTUP.COM driver. If the job controller encounters a fatal error while trying to perform an operation on the system job queue file, it displays an error message on the system console, aborts, and then automatically restarts by creating a new job controller process.

By default, the queue manager (a function of the job controller process) does not automatically restart if the job controller aborts. However, if you specify the /RESTART qualifier with the START/QUEUE/MANAGER command in SYS$MANAGER:SYSTARTUP_V5.COM, the queue manager automatically restarts on recovery from a job controller abort.

This command restores all batch and output queues to the states that existed prior to the interruption of service. The job queue manager file is the same file that was open before the abort. Upon restarting, the job controller uses the default values for the /EXTEND_QUANTITY and /BUFFER_COUNT qualifiers. Previously set values are lost.

Note: **To prevent a looping condition, the job controller does not restart the queue manager if it aborts within two minutes of starting the queue manager.**

If you have to restart the JOB_CONTROL process manually, follow this procedure:

1. Enter the SHOW SYSTEM command to determine the identification numbers of all symbiont processes.
2. Identify all symbiont processes that have a UIC of [1,4] and a name of the following format:

   ```
   SYMBIONT_nnnn
   ```

   Note that user-written symbiont processes may have a process name other than SYMBIONT.
3. Stop symbiont processes by using the following command format:

   ```
   STOP/ID=symbiont_process_id
   ```
4. Execute the STARTUP.COM procedure, specifying the command parameter JOBCTL, as follows:

   ```
   $ @SYS$SYSTEM:STARTUP JOBCTL
   ```

6.3 Managing Queues

As a system manager, you are responsible for setting up and controlling queues. This section introduces the basic DCL commands used to perform routine queue management operations.

The basic commands for creating and controlling queues are as follows:

Command	Description
INITIALIZE/QUEUE	Creates a queue
START/QUEUE	Starts a queue
SET QUEUE	Modifies a queue
STOP/QUEUE	Stops or pauses a queue
SHOW QUEUE	Displays queue attributes

Using qualifiers with these DCL commands allows you to perform the following operations discussed in subsequent sections:

- Initialize and start queues
- Name queues
- Monitor queues
- Modify queues
- Pause queues
- Stop queues
- Restart queue processing
- Delete queues
- Merge output queues
- Restrict access to queues

6.3.1 Initializing and Starting Queues

To set up a queue, you first create or *initialize* the queue, and then start it. Use the following commands to initialize and start a queue:

Command	Description
INITIALIZE/QUEUE	Creates a queue. If the queue is already running, this command has no effect.
START/QUEUE	Starts an initialized queue. If a queue is in a paused state, you can use this command to resume job execution. If a queue is stopped, you can use this command to modify job processing parameters. Jobs listed in the queue and new jobs will execute under the new parameters.
INITIALIZE/QUEUE/START	Initializes and starts a queue. Include this command for each queue created in your SYSTARTUP_V5.COM file.

The INITIALIZE/QUEUE and INITIALIZE/QUEUE/START commands are restricted to users with OPER privilege. The START/QUEUE command requires OPER privilege or EXECUTE (E) access to the queue.

With the INITIALIZE/QUEUE/START command, you initialize and start a queue using a single command. This method is recommended in command procedures (for example, the site-specific startup command procedure SYS$MANAGER:SYSTARTUP_V5.COM) where it is necessary to initialize and immediately start a queue if it is stopped. The following example shows the commands entered in a site-specific startup command procedure to initialize and start the batch queue SYS$BATCH and the printer queues LPA0 and LPB0:

```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=2/BASE_PRIORITY=3  SYS$BATCH
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPA0: QUEUE_A
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPB0: QUEUE_B
```

If the commands are executed when the queue is stopped, the INITIALIZE /QUEUE/START command initializes and starts the queue. Otherwise, if the queue is already running, the command is ignored and no operation occurs. (See the *Guide to Setting Up a VMS System* for a description of startup command procedures.)

Setting up queues is not restricted to startup time. During normal operation, you can create queues as needs dictate. You use the INITIALIZE/QUEUE /START command to create and start a queue. However, separate INITIALIZE/QUEUE and START/QUEUE commands are used to initialize a queue but start it at a later time. For example, suppose you want to initialize a printer queue and make sure that it is set up correctly to process print jobs before you start the queue.

By specifying appropriate qualifiers to the INITIALIZE/QUEUE command, you can assign specific processing attributes to the queue. Once a queue is created, you can also specify additional processing attributes when the queue is started. The qualifiers for specifying different queue attributes are described throughout this chapter. Also, see the *VMS DCL Dictionary* for detailed descriptions of commands and qualifiers.

Use the /DEVICE qualifier to specify the type of output execution queue that you want to initialize or start. If you do not use the /DEVICE qualifier, the default is /DEVICE=printer. Use the /DEVICE qualifier with the INITIALIZE /QUEUE, START/QUEUE or INITIALIZE/QUEUE/START command in the following format:

/DEVICE=(keyword)

The keywords are as follows:

Keyword	Function
Printer	Specifies a printer queue
Terminal	Specifies a terminal queue
Server	Specifies a server queue (use the /PROCESSOR=filename qualifier to specify the user-written symbiont)

6.3.2 Naming a Queue

By default, when you initialize an output execution queue for a particular printer, the queue is assigned the name of the physical device. For example, the following command initializes and starts the queue for the line printer LPA0 and assigns the queue the name LPA0 (the queue name, unlike the device name, has no trailing colon):

```
$ INITIALIZE/QUEUE/START  LPA0:
```

To create an output execution queue and assign it a name that is different from the printer's physical device name, specify the /ON qualifier with the INITIALIZE/QUEUE command as follows:

/ON=[node::]device[:]

The /ON qualifier allows you to assign a queue name that describes the printer's function. For example, you could assign the name LETTER to the queue for a letter quality printer. Users wanting letter-quality printing could then queue their jobs to the queue LETTER.

For naming batch queues, use the /ON=node:: qualifier without specifying a device name. The node name is used only in clustered environments; it must match the node name specified by the SYSGEN parameter SCSNODE for the processor on which the queue executes.

The following command sequence initializes and starts a queue named LETTER, assigns it to the printer LPC0:, and then sends a print job (MY_LETTER.TXT) to the queue.

```
$ INITIALIZE/QUEUE/START/ON=LPC0:  LETTER
$ PRINT/QUEUE=LETTER MY_LETTER.TXT
```

6.3.3 Monitoring Queues

You use the DCL command SHOW QUEUE to monitor the status of queues. To display queue information, enter the SHOW QUEUE command in the following format:

```
$ SHOW QUEUE [/qualifier,...] [queue-name]
```

If you do not specify a qualifier or a queue name, the system displays the status of all queues on the system and all jobs owned by you. The SHOW QUEUE qualifiers allow you to select the type of queue and the amount of information you want to display.

Use the following qualifiers to select the type of queue information you want to display:

Qualifier	Description
/BY_JOB_STATUS	Displays queues that contain jobs of a specified status. You may enter one or more keywords in the following format: [=(executing, holding, pending, retained, timed_release)]. If no keyword is specified, by default the jobs of all status are displayed.
/BATCH	Displays the status of batch execution queues
/DEVICE	Displays the status of output execution queues. You can select a specific type of execution queue by entering one or more keywords in the format [=(printer,terminal,server)]. If no keywords are specified, all types of output queue are displayed.
/GENERIC	Displays the status of generic queues

Use the following qualifiers to select the amount of information you want to display:

Qualifier	Description
/ALL_JOBS	Displays information about all jobs for the selected queue(s).
/BRIEF	Displays a brief listing of information about job entries in the queue. The brief listing is the default when no qualifier is specified with the SHOW QUEUE.
/FULL	Displays complete queue and job information (also displays any ACLs set for the queues).
/SUMMARY	Displays the total number of executing, pending, holding, retained, and timed release jobs.

You can also combine certain qualifiers to further delineate the queue information that you want to display, as shown in the following examples.

This example displays summary information for all printer and terminal queues:

```
$ SHOW QUEUE/SUMMARY/DEVICE=PRINTER,TERMINAL

Generic printer queue CLUSTER_PRINT
    5 holding,  2 timed release

Printer queue HERA_LPB0, on HERA::LPB0, mounted form DEFAULT
    empty

Terminal queue LQ_PRINT, on HERA::TXA7:,
mounted form PORTRAIT_INDENTED (stock=DEFAULT)
    2 pending (8 blocks),   1 executing
```

The next example displays the full status and attributes of all executing jobs.

```
$ SHOW QUEUE/ALL/FULL/BY_JOB_STATUS=EXECUTING
```

```
Batch queue HERA_BATCH, on HERA::
    /BASE_PRIORITY=4 /JOB_LIMIT=10 /OWNER=[1,4] /PROTECTION=(S:E,O:D,G:R,W:W)
    /WSEXTENT=16384 /WSQUOTA=4096

Jobname           Username          Entry       Status
-------           --------          -----       ------
TEST_JOB          PRICE             910         Executing

Submitted 19-May-1988 14:28 /KEEP /LOG=_$333$disk1:[PRICE].LOG; /NOTIFY
/NOPRINT /PRIORITY=100
_$$333$DISK1:[PRICE]TEST_JOB.COM;1 (executing)

Printer queue ZEUS_LPA0, on ZEUS::LPA0, mounted form COLOR (stock=DEFAULT)
     /BASE_PRIORITY=4 /DEFAULT=(Feed, Flag, Form=COLOR, Trailer=one)
     /NOENABLE_GENERIC Lowercase /OWNER=[1,4] /PROTECTION=(S:E,O:D,G:R,W:W)

Jobname           Username          Entry     Blocks    Status
-------           --------          -----     ------    ------
C_MEMO            JONES             918       360       Printing

Submitted 19-May-1988 14:31 /FORM=DEFAULT /PRIORITY=100
_$334$DISK1:[JONES]C_MEMO.LN3;1 (printing)

Server queue M_QUE05, on HERA::, mounted form DEFAULT
    /BASE_PRIORITY=4 /DEFAULT=(FEED, FORM=DEFAULT) /OWNER=[DOC,SMITH]
    /PROCESSOR=MAI$DAEMON /PROTECTION=(S:E,O:D,G:R,W:R) /RETAIN=ERROR

Jobname           Username          Entry     Blocks    Status
-------           --------          -----     ------    ------
NMAIL             SMITH             936       5         Processing
  Submitted 19-MAY-1988 15:05 /FORM=DEFAULT /PRIORITY=100
  _$335$DISK2:[SYS0.NMAIL]NMAIL_33359640384732.WRK;1 (processing)
```

The following list defines the types of queue status returned by the SHOW QUEUE command:

Queue Status	Description
Aligning	The job controller is processing a START/QUEUE/ALIGN command.
Device unavailable	Device to which the symbiont is assigned is not available.
Pausing	The job controller is processing a STOP/QUEUE command.
Paused	A STOP/QUEUE command has been executed.
Resuming	The job controller is processing a START/QUEUE command on a "Paused" queue.
Resetting	The job controller is processing a STOP/QUEUE/RESET command.
Operator service	A PRINT/OPERATOR command has been executed.
Stalled	Symbiont processing temporarily halted due to device related problem.
Stopping	The job controller is processing a STOP/QUEUE[/NEXT /REQUEUE/RESET] command.
Stop pending	Queue will be "Stopped" when current jobs have finished executing.
Stopped	A STOP/QUEUE[/NEXT/REQUEUE/RESET] command has been executed.
Starting	Queue has been started, but the symbiont process is not yet active.

You can further customize the type of queue information that you want to monitor by writing a command procedure that uses the F$GETQUI lexical function. F$GETQUI invokes the $GETQUI system service to return information stored in the system job queue file.

You can use the F$GETQUI lexical function to obtain information about the following types of objects: characteristics, forms, queues, jobs contained in queues, and files of jobs contained in queues. For example, you could write a command procedure to display the total number of blocks of jobs in a pending state in all printer queues. You must have READ access to the job or SYSPRV or OPER privilege to obtain job and file information.

See the *VMS DCL Dictionary* for detailed information on F$GETQUI. For more information about the system service invoked by the F$GETQUI lexical function, refer to the description of the $GETQUI system service in the *VMS System Services Reference Manual.*

6.3.4 Modifying a Queue

Once you initialize a queue, you can change many of its attributes with the DCL command SET QUEUE. This command is restricted to users with OPER privilege or E access to the queue.

The SET QUEUE command allows you to change many queue attributes without having to stop the queue, initialize it, and restart it. For example, the following command modifies the running batch queue, SYS$BATCH:

```
$ SET QUEUE/JOB_LIMIT=4/DISABLE_SWAPPING  SYS$BATCH
```

The command in this example changes the job limit for the queue and disables swapping for all jobs processed in SYS$BATCH. All other attributes of the queue remain the same. The changed attributes do not affect the execution of current jobs; however, all subsequent jobs are executed with the new attributes in effect. See Tables 6–1 and 6–2 for a list of the attributes that can be modified for batch and output queues.

When you modify queue attributes, you should prevent any new jobs from entering the queue. To do so, use the /CLOSE qualifier with the SET QUEUE, INITIALIZE/QUEUE, or START/QUEUE command. The /CLOSE qualifier prevents users from entering jobs in the queue with PRINT or SUBMIT commands. Jobs currently in the queue are not affected. When you have modified the queue attributes, use the /OPEN qualifier to open the queue for incoming jobs.

6.3.5 Pausing a Queue

The DCL command STOP/QUEUE (when used without any qualifiers) temporarily halts the execution of all current jobs in the queue and places the queue in a *paused* state. STOP/QUEUE is restricted to users with OPER privilege or E access to the queue. Pausing an output queue allows you to enter print job positioning and alignment commands to the print symbiont. (See Section 6.4.5 for more information on using the STOP /QUEUE command to control print jobs.)

The DCL command START/QUEUE is used to resume the execution of a paused queue, as described in Section 6.3.7.

6.3.6 Stopping a Queue

To stop a queue, enter one of the following STOP/QUEUE commands, according to the desired stop queue operation:

- **STOP/QUEUE/NEXT**

  This command allows all currently executing jobs to complete and then stops the queue. Once you enter this command, all new jobs are prevented from executing. This command is restricted to users with OPER privilege or E access to the queue.

- **STOP/QUEUE/RESET**

  This command abruptly stops the queue and returns control to the system. Any jobs that are currently executing are stopped immediately. This command is restricted to users with OPER privilege or E access to the queue.

6.3.7 Restarting a Queue

To resume the processing of jobs in a paused or stopped queue, enter the START/QUEUE command in the following format:

START/QUEUE queue-name[:]

This command is restricted to users with OPER privilege or E access to the queue.

Note: **If the STOP/QUEUE/RESET command was entered to halt the queue, you must resubmit all jobs that are not restartable and were executing when the queue was stopped.**

Print jobs are restartable by default; batch jobs are not restartable by default, and must be requeued for processing. To make batch jobs restartable, use the DCL command SUBMIT/RESTART. This command causes a job to restart after a system failure or a requeue operation. (See Section 6.4.3 for information on how to retain a processed job in a queue.)

6.3.8 Deleting a Queue

Perform the following steps to delete a queue:

1 Stop the queue by entering the STOP/QUEUE/NEXT command. (Use STOP/QUEUE/RESET if you want to abort all executing jobs.)

2 Wait for executing jobs to complete.

3 Delete or requeue the entries still pending in the queue.

4 Delete all references to the queue from generic queues by modifying the generic queue. (See Section 6.6.8 for more information on setting up generic output queues.)

5 Delete the queue by entering the DELETE/QUEUE command.

The commands in the following example stop the queue QUE_A, check the contents of the queue, delete the reference to QUE_A from the generic queue SYS_PRINT, and delete the empty queue:

```
$ STOP/QUEUE/NEXT  QUE_A
$ SHOW QUEUE/FULL

Generic printer queue SYS_PRINT
    /GENERIC=(QUE_A,QUE_B) /OWNER=[FIELD,SYSTEM] /PROTECTION=(S:E,O:D,
    G:R,W:W)

Terminal queue QUE_A, stopped, on HECTOR::TXA1:, mounted form DEFAULT
    /BASE_PRIORITY=4 /DEFAULT=(FEED,FORM=DEFAULT) Lowercase
    /OWNER=[FIELD,SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)

Terminal queue QUE_B, on HECTOR::TXA0:, mounted form DEFAULT
    /BASE_PRIORITY=4 /DEFAULT=(FEED,FORM=DEFAULT) Lowercase
    /OWNER=[FIELD,SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)

$ DELETE /QUEUE QUE_A
%JBC-E-REFERENCED, existing references prevent deletion
$ STOP /QUEUE /NEXT SYS_PRINT
$ START /QUEUE SYS_PRINT /GENERIC=QUE_B
$ DELETE/QUEUE  QUE_A
```

The DELETE/QUEUE command is restricted to users with OPER privilege.

6.3.9 Merging Output Queues

When a problem occurs with a print device, you can reroute the queue associated with that device to another queue associated with a functioning device. Use the following procedure to merge two printer queues that are not associated with a generic queue. All commands shown are restricted to users with the OPER privilege or E access to the queues.

1. Stop the queue associated with the malfunctioning print device by using the following command format:

   ```
   $ STOP/QUEUE/NEXT source_queue
   ```

   This command inhibits further queuing but allows the current job to finish processing, unless the print device is inoperable. If the device is inoperable, use the STOP/QUEUE/RESET command to halt the queue and immediately cancel all output to the device.

2. Close the queue from any additional job submissions by entering the /CLOSE qualifier in the following command format:

   ```
   $ SET QUEUE source_queue /CLOSE
   ```

3. Requeue the current job by using the following command format:

   ```
   $ STOP/QUEUE/REQUEUE=target_queue source_queue
   ```

4. Reopen the queue after the entries are requeued and the problem is resolved by entering the following command:

   ```
   $ SET QUEUE source-queue /OPEN
   ```

Note: **If the current print job was originally submitted to a generic queue, the print job is automatically requeued to another device.**

By requeuing the current job, you ensure that it is printed from its last checkpoint. Other jobs in the queue are not scheduled for processing because the queue is stopped.

5. Take the device off line.

6. Reroute existing jobs queued to the malfunctioning print device to another print device by entering the following command:

   ```
   $ ASSIGN/MERGE target_queue source_queue
   ```

   Check to be sure that the attributes of the new print device are appropriate for processing the new jobs.

The following example summarizes the procedure for merging two printer queues that are not associated with a generic queue:

```
$ STOP/QUEUE/NEXT SOURCE_QUEUE
$ STOP/REQUEUE=TARGET_QUEUE SOURCE_QUEUE
$ ASSIGN/MERGE TARGET_QUEUE SOURCE_QUEUE
```

6.3.10 Restricting Access to a Queue

The two security mechanisms for controlling access to queues and jobs in a queue are UIC-based protection, and ACL-based protection. The following sections describes these security features. See the *Guide to VMS System Security* for detailed information on establishing system security.

6.3.10.1 UIC-Based Queue Protection

UIC-based protection restricts the types of jobs and the users who have access to a queue. Operations that apply to queues are controlled by UIC-based protection in the same way that access to other system objects (such as files) is controlled.

When you initialize a queue, the queue is assigned an owner UIC and a protection mask. Jobs are assigned an owner UIC equal to the UIC of the process that submitted the job. Each operation that is performed in a queue or job in a queue is checked against the owner UIC, protection of the queue and the job, and the privileges of the requester.

All operations are checked as follows:

- **Operations that apply to jobs** are checked against the READ (R) and DELETE (D) protection specified for the queue and the owner UIC of the job. In general, R access to a job allows a user to see the attributes of a job, and D access allows the user to delete the job.
- **Operations that apply to queues** are checked against the WRITE (W) and EXECUTE (E) protection specified for the queue and the owner UIC of the queue. A user with W access to a queue can submit jobs to that queue. Users with E access to a queue may act as the operator for that queue, with the ability to affect any jobs in the queue. Users with operator (OPER) privilege have E access to all queues. OPER privilege also enables users to establish queues.

The default UIC-based protection for a queue is (System:E,Owner:D,Group:R,World:W). To find out the protection specified for a particular queue, enter the DCL command SHOW QUEUE/FULL, as shown in the following example:

```
$ SHOW QUEUE/FULL SYS_PRINT

Generic printer queue SYS_PRINT
/GENERIC=(SYS_QUE1,SYS_QUE2) /OWNER=[1,4]
/PROTECTION=(S:E,O:D,G:R,W:W) /RETAIN=ERROR
```

6.3.10.2 ACL-Based Queue Protection

In addition to UIC-based protection, you can also associate access control lists (ACLs) with a queue. ACL-based protection provides a more refined level of protection in cases where certain members of a project group require access to a queue, excluding others of the same UIC group or of other groups.

You use the DCL command SET/ACL to create, modify, or delete ACLs or access control entries (ACEs) established for a queue. Use the following DCL command format:

```
SET ACL/OBJECT_TYPE=QUEUE -
/ACL= (IDENTIFIER= identifier-string,ACCESS= access-code) queue-name
```

The keyword QUEUE specifies that the object type is a batch or output queue. Use the /ACL qualifier to add or modify the ACEs of a particular ACL. The ACL defines the identifiers that a user must hold in order to direct jobs to the queue. It also defines the access code associated with that identifier. You use the Authorize Utility to grant specific ACL identifiers to users. (See the *VMS Authorize Utility Manual* for more information.)

You can use ACLs to restrict queue access to members of a particular project group, as shown in the following example:

```
$ SET QUEUE/PROTECTION=(S,O,G,W,) SYS_QUE1
$ SET ACL/OBJECT_TYPE=QUEUE SYS_QUE1 -
_$ /ACL=(IDENTIFIER=ULTRA_LITE, ACCESS=READ+WRITE+EXECUTE+DELETE) -
_$ /ACL=(IDENTIFIER=MINUTES, ACCESS=READ)
```

The SET QUEUE/PROTECTION command in this example modifies the default UIC-based protection of queue SYS_QUE1 to prevent access by nonprivileged users. The SET ACL command then restricts access to only those members of a project group who hold the ULTRA_LITE identifier. If a queue has ACL protection, the SHOW QUEUE/FULL command displays the ACL information, as shown in the following example:

```
$ SHOW QUEUE/FULL SYS_QUE1
```

```
Batch queue SYS_QUE1, stopped
    /BASE_PRIORITY=4 /JOB_LIMIT=1 /OWNER=[1,4] /PROTECTION=(S:E,O:D,G:R,W:W)
          (IDENTIFIER=[DOC,ULTRA_LITE],ACCESS=READ+WRITE+EXECUTE+DELETE)
          (IDENTIFIER=[XTEAM,MINUTES],ACCESS=READ)
```

See the *Guide to VMS System Security* for detailed information on establishing ACLs for system objects.)

6.3.11 Specifying Queue Characteristics

You can assign specific characteristics to batch or output execution queues with the /CHARACTERISTICS qualifier of the INITIALIZE/QUEUE, START /QUEUE, or SET QUEUE command in the following format:

/CHARACTERISTICS=(characteristic[,...])

Before you associate characteristics with a queue, you must define a characteristic name and assign it a corresponding number using the DCL command DEFINE/CHARACTERISTIC. This command is restricted to users with OPER privilege. The following example defines the characteristic name RED_INK and the corresponding number 3:

```
$ DEFINE/CHARACTERISTIC RED_INK 3
```

You can then use the number 3 or the name RED_INK in subsequent INITIALIZE/QUEUE, START/QUEUE, and SET QUEUE commands to associate those characteristics with a queue.

Note: **Each time you specify queue characteristics with the /CHARACTERISTICS qualifier, all previously set characteristics are erased. Therefore, you must replace the entire list of characteristics each time you enter the command.**

When users include the /CHARACTERISTICS qualifier with a PRINT or SUBMIT command, all the characteristics they specify must also be specified for the queue that will execute the job. If a job is placed in a queue that does not have the characteristics required by that job, the job enters a pending state. The job remains pending until the queue characteristics are modified or the job is deleted with an explicit DELETE/ENTRY command. Users need not specify all characteristics of a queue when they enter a PRINT or SUBMIT command, as long as the characteristics specified are a subset of those established for the queue.

By default, a queue or job has no characteristics defined. To determine the characteristics defined for a queue, enter the DCL command SHOW QUEUE/FULL.

To delete a systemwide queue characteristic definition, enter the DCL command DELETE/CHARACTERISTIC, for example:

```
$ DELETE/CHARACTERISTIC RED_INK 3
```

6.4 Managing Jobs

As system manager, you are responsible for controlling the flow of batch and print jobs and for maintaining efficient job processing performance. Some of the routine tasks for managing jobs include the following:

- Monitoring job processing
- Handling error conditions that require deleting, retaining, or requeuing jobs
- Modifying job processing attributes
- Controlling print job position and alignment

You perform these tasks by entering the DCL commands described in the sections that follow.

The system queue manager (a function of the job controller process) schedules print jobs in the following order, according to specific job processing attributes:

1. Stock type

   The system checks the job's paper stock attribute, to see if it matches the stock mounted on the queue. If the stocks are NOT identical, the job is placed in a pending state.

2. Priority level

   The system checks the job's scheduling priority. The job with the highest priority value is processed first.

3. Size

   Job size is checked. Among jobs of identical priority, the smallest sized job is processed first.

4. Submission time

   If all other job attributes are identical, the job that was submitted first is processed first.

Note: **Batch jobs are scheduled by priority and submission time only.**

To display the attributes of a job, use the DCL command SHOW ENTRY, described in the next section. Also, see Section 6.4.4 for information on modifying job processing attributes.

6.4.1 Monitoring Jobs

You use the DCL command SHOW ENTRY to monitor the status of batch and print jobs. Unlike SHOW QUEUE, which displays the status of all queues on the system, SHOW ENTRY has the advantage of displaying the status of a specific job. Use the following format to specify the SHOW ENTRY command:

```
$ SHOW ENTRY [entry-number[,...]]
```

If you do not specify an entry number, the system displays all jobs owned by you (or by the user specified with the /USER qualifier). (You must have a privileged account to display jobs owned by users outside of your UIC group.) You can also display a group of jobs by entering a list of entry numbers on the command line. The SHOW ENTRY qualifiers allow you to specify the type of job information that you want to display. The following command displays jobs owned by user PRICE:

```
$ SHOW ENTRY/USER=PRICE

Jobname         Username        Entry  Blocks     Status
-------         --------        -----  ------     ------
CMEMO           PRICE           101    45         Printing

On terminal queue LASER$PRINT
```

In addition to job status information, the /FULL qualifier also displays the time the job was submitted, the file specification, and the job processing attributes, for example:

```
$ SHOW ENTRY/BATCH/FULL

Jobname         Username        Entry  Blocks     Status
-------         --------        -----  ------     ------
TESTBUILD       PRICE           102    106        Holding

  On generic batch queue CLUSTER_BATCH

  Submitted 19-DEC-1988 11:19 /NOTIFY  /PRIORITY=100
_$DISK$2:[PRICE]TESTBUILD.COM;5
```

For a complete description of the SHOW ENTRY command and qualifiers, see the *VMS DCL Dictionary*.

The following list describes the types of job status returned by the SHOW ENTRY command:

Status	Description
Aborting	Executing job is terminating
Executing	Job is executing from a batch queue
Holding	Job is being held until explicitly released
Holding until	Job is being held until a specified time
Pending	Job is in a waiting state
Printing	Job is executing from a printer or terminal queue
Processing	Job is executing from a server queue
Retained on Completion	Job remains in the queue upon completion
Retained on Error	Job remains in the queue upon encountering an error
Waiting	Symbiont refuses the job

6.4.2 Deleting a Job

Under certain circumstances, it is necessary to terminate an executing batch or print job. For example, you may need to terminate a program that has entered an endless loop or a job that is executing on a faulty print device.

Follow this procedure to delete either a pending or an executing job:

1 Determine the entry number of the job and the name of the queue by entering the command:

```
SHOW ENTRY/USER=username
```

2 Delete the job by entering the following command:

```
DELETE/ENTRY=(entry-number[,...]) [queue-name[:]]
```

The DELETE/ENTRY command is restricted to users with either OPER privilege, E access to the queue, or D access to the specified job.

Assume a user has observed that a job is a program processing in an endless loop. The user is not the owner of the job and lacks sufficient privilege to stop it. The user enlists your aid as the system manager. You delete the job by entering the command DELETE/ENTRY=entry_number. If you do not know the queue name or entry number, enter the command SHOW QUEUE /BATCH/ALL/BY_JOB_STATUS=EXECUTING, and then delete the job, for example:

```
$ SHOW QUEUE/BATCH/ALL/BY_JOB_STATUS=EXECUTING

Batch queue ZEUS_BATCH, on ZEUS

 Jobname         Username     Entry  Status
 -------         --------     -----  ------
 2307SMRCL       MARCO         1719  Executing
 TEST            JONES         1720  Executing

$ DELETE/ENTRY=1719
```

6.4.3 Retaining Jobs in a Queue

To retain a job in a queue after it has been processed, specify the /RETAIN qualifier with the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. The /RETAIN qualifier has the following format:

/[NO]RETAIN[=option]

You indicate the type of job that the queue is to retain by specifying one of the following options:

Option	Description
ALL	Specifies that jobs be retained in the queue in a completed status after they are executed
ERROR	Specifies that jobs be retained only if they completed unsuccessfully (the completion status has the low bit clear)

By default, jobs are **NOT** retained.

The following command specifies that the queue retain all jobs that complete with a "nonsuccess" (low bit clear) status:

```
$ SET QUEUE/RETAIN=ERROR  BATCH_QUE
```

6.4.4 Modifying Job Processing Attributes

You can modify certain job processing attributes by specifying qualifiers with the command SET ENTRY entry-number, as shown in the following table:

Qualifier	Description
/[NO]AFTER	Controls whether a job is held until after a specified time
/[NO]HOLD	Controls whether a job is available for immediate processing or held until it is released for processing
/[NO]PASSALL	Specifies whether the symbiont bypasses all formatting and sends the output QIO to the driver with format suppressed (see Section 6.6.5 for information on the /[NO]PASSALL qualifier)
/PRIORITY	Specifies the scheduling priority of the job
/RELEASE	Releases a previously held job
/REQUEUE	Requests that the job be moved from the original queue to the specified queue; this qualifier can also be used with the STOP/QUEUE/ENTRY command
/SETUP	Calls for the specified modules from the device control library (see Section 6.6.6 for information on device control libraries)

(See the *VMS DCL Dictionary* for complete descriptions of these qualifiers.)

6.4.4.1 Holding and Releasing a Job

The /HOLD qualifier of the SET ENTRY command controls whether a job is to be made available for immediate processing. You release a held job by using either the /NOHOLD or the /RELEASE qualifier.

To request that the job be held until after a specified time, use the /AFTER qualifier with the command SET ENTRY. The job is queued for immediate processing when the specified time arrives. The /AFTER=time qualifier accepts either absolute or delta time values in the format [dd-mmm-yyyy] [hh:mm:ss.cc]. You can also specify the following keywords:

TODAY
YESTERDAY
TOMORROW

The following command holds a print job until it is queued for processing at the specified date and time:

```
$ SET ENTRY 1121/AFTER=12-JUL-1988:17:30
```

You can use the /NOAFTER qualifier to immediately release a job that is being held until after a specified time.

The /RELEASE qualifier releases a job that is being held for any of the following reasons:

- A job was submitted with the /HOLD qualifier
- A completed job was held in a queue by the /RETAIN qualifier
- A user-written symbiont has refused a job
- A job was submitted with the /AFTER qualifier

This example shows how to hold and release a batch job:

```
$ STOP/QUEUE/ENTRY=1234/HOLD
```

```
$ SET ENTRY 1234/RELEASE
```

6.4.4.2 Requeuing a Job

To stop and requeue the executing print job, enter the STOP/QUEUE /REQUEUE command. This command suspends the currently executing job and requeues it to the specified queue, for example:

```
$ STOP/QUEUE/REQUEUE=ALPHA_LPA0 ALPHA_LPB0
```

This command causes the executing print job on ALPHA_LPB0 to be stopped and requeued to ALPHA_LPA0. The queue does not stop; only the currently executing job is affected. Other jobs remain pending in the queue until they are processed.

You can further qualify the STOP/QUEUE/REQUEUE command with the /HOLD qualifier. To hold an aborted print job, enter the STOP/QUEUE /REQUEUE/HOLD command in the following format:

STOP/QUEUE/REQUEUE/HOLD [queue-name]

When you specify /HOLD, the aborted job is placed in a hold state for later release with the SET ENTRY/RELEASE command. If you do not need to process a job that is being held in a queue, you can delete the job with the DELETE/ENTRY command.

Note: If you are requeuing a job on a batch queue, you must include the /ENTRY=n qualifier, for example:

```
$ STOP/QUEUE/ENTRY=1251/REQUEUE=FRED_BATCH
```

6.4.4.3 Changing the Scheduling Priority of a Job

You can change the scheduling priority of a job by using the /PRIORITY=n qualifier with the SET ENTRY command. Do not confuse the job scheduling priority with the base priority of a queue.

The job scheduling priority value must be in a range of 0 through 255, where 0 is the lowest priority and 255 is the highest. The default value for /PRIORITY is the value of the SYSGEN parameter DEFQUEPRI (usually set at 100). You must have either OPER or ALTPRI privilege to raise the priority value above the value of the SYSGEN parameter MAXQUEPRI. No privilege is needed to set the priority of your own job lower than the MAXQUEPRI value. The following example changes the priority of a job to 50:

```
$ SET ENTRY 1131/PRIORITY=50
```

6.4.5 Controlling Print Job Position and Alignment

Pausing an output queue allows you to communicate with the print symbiont interactively. You enter the STOP/QUEUE command (without any qualifiers) to pause a queue. Once a queue is paused, you can perform operations such as the following:

- Specify the position at which a suspended job is to resume printing
- Specify the number of pages and the type of data for aligning printer forms

These operations are described in the sections that follow.

6.4.5.1 Specifying the Position of Print

To specify the position at which the current job is to resume printing, pause the queue, then enter the START/QUEUE command with any of the following qualifiers:

Qualifier	Description
/[NO]BACKWARD[=pages]	Specifies the number of pages the file is to be backspaced before printing is resumed. A value of 1 indicates that printing is to resume at the top of the current page. If you omit the page value, the file is backspaced one page.
/[NO]FORWARD[=pages]	Specifies the number of pages the file is to be forward-spaced before printing is resumed. A value of 1 indicates that printing is to resume at the top of the next page. If you omit the page value, the file is forward-spaced one page.
/[NO]SEARCH=string	Specifies that printing is to resume at the page containing a particular search string. The search for the string moves forward, beginning on the page following the current page. During the search, consecutive tabs and spaces are treated as a single space, and character case is ignored.
/TOP_OF_FILE	Specifies that printing is to resume at the beginning of the file.

When you must use more than one positioning qualifier with the same START /QUEUE command, file positioning is performed in the following order:

/TOP
/FORWARD
/BACKWARD
/SEARCH

The command in the following example resumes output on printer LPA0 after advancing 15 pages from the beginning of the file.

```
$ START/QUEUE/TOP/FORWARD=15 SYS_LPA0
```

6.4.5.2 Aligning Printer Forms

To print alignment data to aid in aligning printer forms, enter the START /QUEUE command along with the /ALIGN qualifier:

```
/ALIGN[=(option[,...])]
```

The following options control the number of alignment pages and type of alignment data:

Option	Description
MASK	Specifies that input data is masked by replacing alphabetic characters with the character X and numeric characters with the number 9. Mask characters allow you to prevent the printing of sensitive information. If you omit the MASK option, data is printed unaltered.
n	A decimal number in the range 1 through 20 that specifies the number of alignment pages to print. By default, one page of alignment data is printed.

You can use the /ALIGN qualifier along with any of the file positioning qualifiers described in the previous section. File positioning is performed before alignment data is printed. After the alignment is complete, the queue enters a paused state until you restart it by reentering the START/QUEUE command. Printing resumes from the point that alignment data started; that is, the task is backspaced over the pages printed for alignment.

The command in the following example requests masked alignment for four pages of output.

```
$ START/QUEUE/BACKWARD=2/ALIGN=(MASK,4) SYS_LPA0
```

In this example, the file for the job that was being printed when the queue was paused is backspaced two pages before alignment is performed. Four pages of alignment mask characters are printed. Then the output for the current job is positioned backward four pages, and the queue pauses.

6.5 Procedures Specific to Batch Queues

A batch execution queue controls the execution of batch jobs, and defines default system resources for jobs that execute on it. You must decide on the number of batch queues for your installation and determine attributes such as the following:

- Job limit
- Base priority
- Swap mode
- CPU limits
- Working set quotas and limits

(See the *Guide to Setting Up a VMS System* for a discussion of limits, quotas, and priorities.)

Before users can submit batch jobs, the system manager must create at least one batch queue. Jobs do not begin processing until the system manager starts the batch queue. (Section 6.3.1 discusses how to create and start a queue.) More than one batch job can execute at the same time from a single batch queue. The number of jobs that can execute simultaneously depends on the job limit that you establish for the queue.

After a batch queue has been started, the job with the highest relative priority is the first candidate for execution. For jobs having the same priority, the earliest submitted job is the first candidate for execution.

Encourage users to submit large jobs (such as compiling and linking large programs) as batch jobs and to reserve interactive use of the system for jobs that do not require extensive resources. Toward this end, you may do the following:

- Restrict the working set size of interactive jobs by providing a small (for example, 300) WSEXTENT value in the UAF records. (See the *Guide to Setting Up a VMS System* for more information on the WSEXTENT field of the UAF record.)
- Expand the working set size of batch jobs by providing a large (for example, 500) WSEXTENT value when you create the batch queue.

You can likewise restrict and expand time limits on jobs by setting the CPU values. Sections 6.5.2, 6.5.3, and 6.5.4 provide some guidelines for setting up batch execution queues in different environments.

You control the attributes of a batch queue by entering specific qualifiers with the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. Table 6–1 gives a summary of the attributes that can be associated with a batch queue. See the *VMS DCL Dictionary* for complete details on the qualifiers described in Table 6–1.

Table 6–1 Specifying Attributes for Batch Execution Queues

Qualifier	Description
/[NO]BATCH	Indicates that you are initializing a batch queue. The default is /NOBATCH. For an existing queue, you can use the /BATCH qualifier only if the queue is already a batch queue.
/BASE_PRIORITY	Specifies the base process priority at which jobs in the queue are processed.
/[NO]CHARACTERISTICS	Specifies one or more characteristics for processing jobs in a queue. The default is /NOCHARACTERISTICS. A characteristic can be either a value from 0 through 127 or a characteristic name that has been defined by the DEFINE/CHARACTERISTIC command.
/CLOSE	Prevents any new jobs from being entered in the queue through PRINT or SUBMIT commands or as the result of a requeue operation.
/CPUDEFAULT	Defines the base queue value for the default CPU time limit for batch jobs executed in this queue. The time cannot exceed the CPU time limit set by the /CPUMAXIMUM qualifier.
/CPUMAXIMUM	Defines the maximum CPU time for batch jobs. Use this qualifier to override the CPU time limit specified in the UAF file.
/[NO]DISABLE_SWAPPING	Controls whether batch jobs executed from a queue can be swapped in and out of memory. The default is /NODISABLE_SWAPPING.
/[NO]ENABLE_GENERIC	Specifies whether jobs queued to a generic queue can be placed in this queue for processing. The default is /ENABLE_GENERIC.
/JOB_LIMIT	Indicates the number of batch jobs that can be executed concurrently from a batch queue.

Table 6–1 (Cont.) Specifying Attributes for Batch Execution Queues

Qualifier	Description
/ON=node::	Specifies the node on which this batch execution queue is located. The node name is used only in clustered environments; it must match the node name specified by the SYSGEN parameter SCSNODE for the processor on which the queue executes. See Section 6.3.2 for more information on naming queues.
/OPEN	Allows jobs to be entered in a queue through PRINT or SUBMIT commands or as the result of a requeue operation.
/OWNER_UIC	Enables you to change the user identification code (UIC) for the queue.
/PROTECTION	Specifies the protection of the queue. By default, the queue protection is (SYSTEM:E, OWNER:D, GROUP:R, WORLD:W).
/[NO]RETAIN	Specifies that jobs be retained in the queue in a completed status after they have executed. The default is /NORETAIN.
/WSDEFAULT	Defines the base queue value for the working set default for batch jobs executed in this queue. The value set by this qualifier overrides the value defined in the UAF of any user submitting a job to the queue.
/WSEXTENT	Defines the base queue value for the working set extent for batch jobs executed in this queue. The value set by this qualifier overrides the value defined in the UAF of any user submitting a job to the queue.
/WSQUOTA	Defines the base queue value for the working set page size for batch jobs executed in this queue. The value set by this qualifier overrides the value defined in the UAF of any user submitting a job to the queue.

6.5.1 Overriding Batch Queue Attributes

You submit batch jobs to the VMS operating system and queue them for execution in two ways:

- As command procedure files submitted by entering the DCL command SUBMIT. These files are placed in a batch queue and selected for execution according to their relative priority in the queue. By default, the name of this batch queue is SYS$BATCH.
- As batch job files submitted by entering the DCL command JOB from a card reader. These batch job files are spooled onto a disk and placed in a batch queue. Unless the JOB command specifies otherwise, the name of this batch queue is SYS$BATCH.

When a job executes on a batch queue, the job is assigned the attributes of the queue. Users, however, can explicitly override some queue attributes for a particular job by specifying qualifiers to the DCL command SUBMIT. The following qualifiers are useful for balancing the system resources for processing batch jobs:

Qualifier	Description
/AFTER=time	Specifies that the job be held until after a designated time. If the designated time has passed or the /NOAFTER qualifier is specified, the job is queued for immediate processing.
/CLI	Enables you to specify a different command language interpreter (CLI) to use in processing the job.
/CPUTIME	Defines a CPU time limit for the batch job. When a job needs less CPU time than authorized, use the /CPUTIME qualifier to override the base value established for the queue or authorized in your UAF.
/PRIORITY	Specifies the job scheduling priority for a job. You must have either OPER or ALTPRI privilege to raise the priority above the value of the SYSGEN parameter MAXQUEPRI. No privilege is needed to set the priority lower than the MAXQUEPRI value. The default priority of a queue is determined by the SYSGEN parameter, DEFQUEPRI.
/WSDEFAULT	Defines a working set default for a batch job. Use this qualifier to override the base value established for the queue.
/WSEXTENT	Defines a working set extent for a batch job. You cannot request a higher value than your default.

See the *VMS DCL Dictionary* for complete descriptions of the SUBMIT command qualifiers.

You can also control certain attributes of batch jobs that are submitted as command procedure files by entering the following SET commands:

Command	Description
SET OUTPUT_RATE=time	Controls the rate at which output is written to a batch job log file. By default the contents of the buffer is written to a log file once per minute.
SET RESTART_VALUE=string	Defines the value of the symbol BATCH$RESTART. This symbol can be used to control the point at which a command procedure restarts after encountering a system interruption. BATCH$RESTART is the value of the most recently executed SET RESTART_VALUE command. The $RESTART symbol is a Boolean value that is true if a system interruption has occurred.
SET ENTRY/NOCHECKPOINT	Erases the value established by the most recently executed SET RESTART_VALUE command by clearing BATCH$RESTART.

See the *VMS DCL Dictionary* for detailed descriptions of these SET commands.

6.5.2 Setting Up Batch Queues on Small Systems

On small systems (under 2MB), you should add up the pages required for the batch working sets and ensure that enough fluid memory remains for interactive jobs. Fluid memory can be reassigned from one process to another through swapping and paging. (You can calculate fluid memory as the space that remains when you subtract the number of pages permanently allocated to the VMS operating system from the total memory. To obtain these values, enter the DCL command SHOW MEMORY.)

If fluid memory drops below the value of the WSMAX system parameter, you must reduce the number of batch jobs or make the FAST and SLOW jobs swappable. Otherwise, a deadlock could result. (See Section 6.5.4 for information on setting up FAST and SLOW queues.)

6.5.3 Setting Up Batch Queues on Interactive Systems

This section provides guidelines for setting up a batch queue for a system that is predominantly interactive. The values in this section are typical for a VAX-11/780 system. The values should be adjusted according to the type of CPU and the amount of memory available on your system.

1. Set up one batch queue named SYS$BATCH, the name of the default batch queue.
2. Give SYS$BATCH the following characteristics:
   - Job limit—1 to 4
   - Base priority—3 or 4 (one less than SYSGEN parameter DEFPRI)
   - Swapping mode—swapping enabled (default)
   - Working set default—200 to 300

- Working set quota—300 to 500
- Working set extent—400 to 2000 (depending on memory size; also, value should be greater than working set quota)
- CPU default—infinite (default)
- CPU maximum—infinite (default)

Normally, you would include the following command in your site-specific startup command procedure to create and start a queue on a VAX–11/780 with 4MB of memory:

```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=2/BASE_PRIORITY=3 -
_$ /WSDEFAULT=300/WSQUOTA=500/WSEXTENT=2000  SYS$BATCH
```

6.5.4 Setting Up Batch Queues on Batch Systems

The following guidelines are useful for setting up batch queues for a system that is predominantly a batch system and in which editing is the principal interactive activity:

1. Set up three batch queues as follows:
   - SYS$BATCH—the default batch queue with swapping enabled.
   - FAST—a queue for executing high-priority jobs that should not be swapped out of memory.
   - SLOW—a "background" queue for processing low-priority jobs. Typically, these are large jobs with large requirements for physical memory. Usually, it is not economical to swap such jobs out of memory. You can adjust the system workload by stopping and restarting background queues as needed.
2. Give SYS$BATCH the following characteristics:
   - Job limit—6 to 10
   - Base priority—4 (by default)
   - Swapping mode—swapping enabled (by default)
3. Give FAST the following characteristics:
   - Job limit—1 (by default)
   - Base priority—4 (by default)
   - Swapping mode—swapping disabled
4. Give SLOW the following characteristics:
   - Job limit—1 (by default)
   - Base priority—low (3, for example)
   - Swapping mode—swapping disabled

Normally, you would include the following commands in your site-specific startup command procedure to create and start these queues:

```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=6  SYS$BATCH
$ INITIALIZE/QUEUE/START/BATCH/DISABLE_SWAPPING  FAST
$ INITIALIZE/QUEUE/START/BATCH/BASE_PRIORITY=3 -
_$ /DISABLE_SWAPPING  SLOW
```

6.5.5 Setting Up Generic Batch Queues

Unlike an output queue, a batch queue can execute more than one job at a time. For this reason, it is often not necessary on a single-node system to create more than one batch queue of the same type and assign them to a generic batch queue.

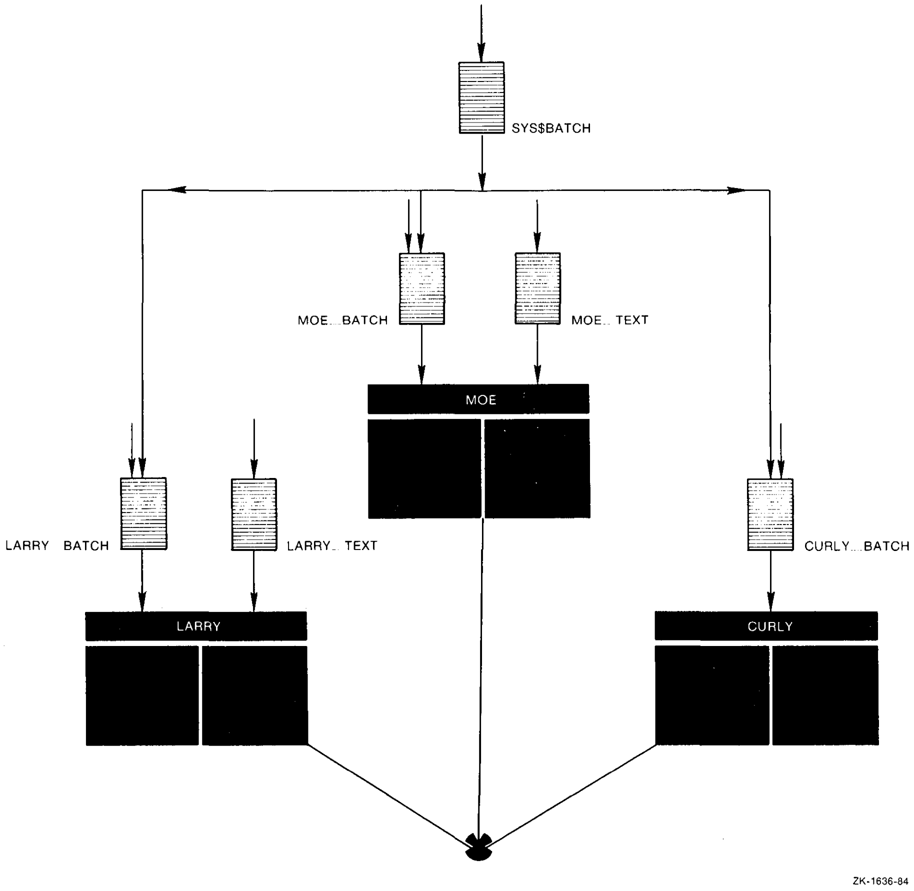
On a cluster, however, you distribute batch processing across the nodes to balance the use of processing resources. To achieve this workload distribution, assign local batch queues to one or more clusterwide generic batch queues. Managers establish and maintain the clusterwide queues with the same commands used to manage queues on a single-node system.

Instead of having a queue named SYS$BATCH set up on each cluster node, you create a clusterwide generic batch queue and name it SYS$BATCH.

For example, in Figure 6–1 batch queues from each node are assigned to a clusterwide generic batch queue named SYS$BATCH. Users can submit a job to a specific queue, or if they have no special preference, submit it by default to the clusterwide generic queue, SYS$BATCH. The generic queue in turn places the job in an available assigned queue in the cluster. If more than one assigned queue is available, the system selects the queue that minimizes the ratio (executing jobs/job limit) for all assigned queues.

All clusterwide queues are controlled by a single, cluster-common job controller queue file. This file makes queues available across the cluster and enables jobs to execute on any queue from any node, provided that the necessary mass storage volumes can be accessed by the node on which the job executes.

Figure 6–1 Batch Queue Configuration With Clusterwide Generic Queue



The commands in the following example summarize how to set up generic queues on a cluster:

1 Start the job queue manager:

```
$ START/QUEUE/MANAGER [file-spec]
```

The default file spec is SYS$SYSTEM:JBCSYSQUE.DAT.

2 Set up printer queues:

```
$ INITIALIZE/QUEUE/ON=node_name::device: QUEUE1
$ INITIALIZE/QUEUE/ON=node_name::device:/START  QUEUE2
```

3 Set up generic printer queues:

```
$ INITIALIZE/QUEUE/GENERIC=(QUEUE1,QUEUE2...)/START SYS_PRINT
```

4 Set up batch queues:

```
$ INITIALIZE/QUEUE/BATCH/ON=node_name:: QUEUE3

$ INITIALIZE/QUEUE/BATCH/ON=node_name:: /START QUEUE4
```

5 Set up generic batch queues:

```
$ INITIALIZE/QUEUE/BATCH/GENERIC=(queue3,queue4...)/START SYS_BATCH
```

See the *VMS VAXcluster Manual* for detailed information on setting up queues in a clustered environment.

6.6 Procedures Specific to Output Queues

An output execution queue controls the processing of print jobs by an independent symbiont process that is associated with the queue. Before users can submit print jobs, the system manager must initialize at least one output queue. Jobs do not begin processing until the system manager starts the queue. (Section 6.3.1 describes how to initialize and start a queue.)

Print jobs are queued for processing in one of two ways:

- **Implicitly**—without the direct intervention of a user
- **Explicitly**—with the direct intervention of a user

Implicit Printing

Print jobs are queued implicitly when a process directs its output to a spooled device. The output is then placed in a temporary file. When the file is closed, it is submitted to an output queue. Both the spooling of the output file to an intermediate device and the subsequent queuing of a job consisting of this file occur without the direct intervention of a user. Implicit printing uses default attributes and bypasses the user's ability to specify job attributes on the command line. Section 6.6.9 discusses spooled devices in detail.

Explicit Printing

Print jobs are queued explicitly when users enter the PRINT command. Users can explicitly queue one or more files (from a disk) for printing. (The *VMS DCL Dictionary* describes the PRINT command and available qualifiers in detail.) The files specified in the PRINT command are queued as a print job. If several files make up a print job, they are printed together.

Unlike batch execution queues, an output execution queue can print only one job at a time. Print jobs are scheduled for printing according to their stock type, priority, size, and submission time (as described in Section 6.4).

By default, print jobs queued by entering the PRINT command are placed in the queue named SYS$PRINT. Thus, to use the default version of the PRINT command on a system with only one printer, name the printer queue SYS$PRINT in your site-specific startup command procedure as follows:

```
$ INITIALIZE/QUEUE/START/ON=LPAO:  SYS$PRINT
```

To use the default version of the PRINT command on a system with several line printers of matching characteristics, you would usually establish SYS$PRINT as the name of a generic queue. Section 6.6.8 describes how to establish generic output queues.

You control the attributes of an output queue by entering specific qualifiers to the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. Table 6–2 gives a summary of the attributes that can be associated with output queues. (See the *VMS DCL Dictionary* for complete descriptions of these qualifiers.)

Table 6–2 Specifying Attributes for Output Execution Queues

Qualifier	Description
/BASE_PRIORITY	Specifies the base process priority at which the symbiont serving this queue will execute. You must stop and restart all output queues sharing the symbiont to change the symbiont base process priority.
/[NO]BLOCK_LIMIT	Limits the size of print jobs that can be executed on an output queue. The default is /NOBLOCK_LIMIT. You can use this qualifier to reserve certain printers for certain size jobs.
/[NO]CHARACTERISTICS	Specifies one or more characteristics for processing jobs in a queue. The default is /NOCHARACTERISTICS. A characteristic can be either a value from 0 through 127 or a characteristic name that has been defined by the DEFINE/CHARACTERISTIC command.
/CLOSE	Prevents any new jobs from being entered in the queue through PRINT or SUBMIT commands or as the result of a requeue operation.
/[NO]DEFAULT	Establishes defaults for certain options of the PRINT command. The default is /NODEFAULT. /DEFAULT options include FLAG, BURST, TRAILER, FEED, and FORM=type. Once an option is set for the queue by the /DEFAULT qualifier, users do not have to specify that option in their PRINT commands. However, users may override these options by including them on the DCL PRINT command line.
/[NO]ENABLE_GENERIC	Specifies whether files queued to a generic queue can be placed in this queue for processing. The default is /ENABLE_GENERIC.
/FORM_MOUNTED	Specifies the form mounted on the output queue. You specify the form by using a numeric value or a form name that has been defined by the DEFINE/FORM command. If the stock of the form associated with the job is not identical to the stock of the form mounted on the queue, the job enters a pending state.
/[NO]LIBRARY	Specifies the file name for the device control library. When you are initializing an output queue, you can use this qualifier to specify an alternate device control library. The default library is SYS$LIBRARY:SYSDEVCTL.TLB.

Table 6–2 (Cont.) Specifying Attributes for Output Execution Queues

Qualifier	Description
/ON=device[:]	Specifies the device on which this execution queue is located. (Specify the node name without the device name for clustered environments).
/OPEN	Allows jobs to be entered in a queue through PRINT or SUBMIT commands or as the result of a requeue operation.
/OWNER_UIC	Enables you to change the user identification code (UIC) for the queue.
/[NO]PROCESSOR	Allows users to specify their own print symbiont image. The file name specifier can be any valid file name. It specifies that the symbiont image to be executed is SYS$SYSTEM:file-name.EXE. By default, SYS$SYSTEM:PRTSMB.EXE is executed. The /NOPROCESSOR qualifier cancels the effect of a previous /PROCESSOR setting.
/PROTECTION	Specifies the protection of the queue. By default, the queue protection is (SYSTEM:E, OWNER:D, GROUP:R, WORLD:W).
/[NO]RECORD_BLOCKING	Determines whether the symbiont can block together output records for transmission to the output device. The default is /RECORD_BLOCKING.
/[NO]RETAIN	Specifies that jobs be retained in the queue in a completed status after they have executed. The default is /NORETAIN.
/SCHEDULE=[NO]SIZE	Specifies whether pending jobs in a printer, terminal, or server queue are scheduled for printing based on the size of the job. When the default /SCHEDULE=SIZE is in effect, smaller jobs print before larger ones.
/[NO]SEPARATE	Specifies the job separation attributes for a printer or terminal queue. Users cannot override these options. The default is /NOSEPARATE. Job separation options include BURST, FLAG, TRAILER, and RESET=module. The selected options establish mandatory attributes for all jobs processed in the queue.
/WSDEFAULT	Defines a working set default for the symbiont process. You must stop and restart all output queues to change the symbiont base process priority.

Table 6–2 (Cont.) Specifying Attributes for Output Execution Queues

Qualifier	Description
/WSEXTENT	Defines a working set extent for the symbiont process. You must stop and restart all output queues to change the symbiont base process priority.
/WSQUOTA	Defines the working set page size for the symbiont process. You must stop and restart all output queues to change the symbiont base process priority.

6.6.1 Controlling Jobs by Size

You can impose limits on the size of jobs that are accepted by a queue by specifying the /BLOCK_LIMIT qualifier with the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. The /BLOCK_LIMIT qualifier has the following format:

/[NO]BLOCK_LIMIT=([lower][,upper])

Limiting the size of jobs in the queues allows you to control the flow of jobs and make efficient use of the printers. The /BLOCK_LIMIT qualifier allows you to limit processing on a queue to either small or large jobs. By default, there are no limits on file size for a queue. To impose block size limits, you must specify at least one of the following parameters:

Parameter	Description
lowlim	A decimal number referring to the minimum number of blocks that can be scheduled for processing by the queue. If a print job is submitted that contains fewer blocks than the **lowlim** value, the job remains pending until the block limit for the queue is changed. By default there is no lower limit on job size.
uplim	A decimal number referring to the maximum number of blocks that can be scheduled for processing by the queue. If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed. By default there is no upper limit on job size.

(For a complete description of the /BLOCK_LIMIT qualifier, see the *VMS DCL Dictionary*.)

In the following example, processing on a printer queue LARGE is limited to jobs of more than 500 blocks:

```
$ INITIALIZE/QUEUE/ON=LPB0:/BLOCK_LIMIT=(500,"") LARGE
```

You can set up a generic queue to feed two execution queues that restrict jobs by size. One execution queue could handle small jobs, and one could handle large jobs, as shown in the following example:

```
$ INITIALIZE/QUEUE/ON=LPB0:/BLOCK_LIMIT=(500,"") LARGE
$ INITIALIZE/QUEUE/ON=LPC0:/BLOCK_LIMIT=(1,500) SMALL
```

Note: **If you specify a single value for *n* in the qualifier /BLOCK_LIMIT=n, the value is interpreted as the upper limit for that queue.**

6.6.2 Using Forms to Control Print Jobs

Print forms serve the following two functions:

- Forms determine certain page formatting attributes (such as margins and page length).
- The paper stock attribute specified in the form definition is used in scheduling print jobs.

The print symbiont uses input from the form to format the pages and associate other processing attributes with a job when it is executed. A form can include any of the following attributes: paper stock, width, length, truncate, wrap, margin, page setup, form setup, sheet feed, and description. You use the command DEFINE/FORM and its qualifiers to create a form and determine its attributes. See Section 6.6.2.1 for more information on defining a form.

To understand how forms are used in print job scheduling, you must first understand how a print form is associated with a job. When you submit a print job with the DCL command PRINT/FORM=type, the specified form (and its stock type) are associated with the job. The paper stock specified for the job becomes the current paper stock mounted on the queue, overriding any default form (and its stock type) associated with the queue.

If you submit a print job without the /FORM=type qualifier, the job is processed with the default form for the queue. Section 6.6.2.3 describes how to establish a queue-specific default form for an execution queue.

DIGITAL supplies a systemwide default form (named DEFAULT) for all queues on a VMS operating system. If you do not establish a queue-specific default form, the queue has the systemwide default form. Section 6.6.2.4 describes how to modify the systemwide default form.

The system job queue manager checks the stock attributes when it schedules a job for printing. If the stock types match, the job can be executed; if they don't match, the job remains pending in the queue until you change the mounted form. Use the DCL command SET QUEUE/FORM_MOUNTED=type to change the form mounted on a queue.

Entering the command SHOW QUEUE/FORM/FULL, displays the form and the stock type associated with a queue, as shown in the following example:

```
$ SHOW QUEUE/FORM/FULL

Form name                    Number      Description

DEFAULT                          0       System-defined default
    /LENGTH=66     /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /TRUNCATE /WIDTH=132

PORTRAIT (stock=DEFAULT)       106       80 by 60 (portrait)
    /LENGTH=60    /SETUP=(PORTRAIT) /STOCK=DEFAULT  /WIDTH=80

LETTER (stock=DEFAULT)         110       LN03 indented memo format
   /LENGTH=64  /MARGIN=(TOP=2,LEFT=5) /STOCK=DEFAULT /TRUNCATE /WIDTH=80
```

If the stock currently mounted on the queue (displayed in the /STOCK=type field in the previous example) does not match the stock of the default form for the queue (displayed in the PORTRAIT (stock=type) field), then any jobs submitted without an explicit form in the PRINT command will not execute. See Section 6.6.2.2 for more information on mounting a form on a queue.

The concepts for using forms to control print jobs are elaborated on in the following sections.

6.6.2.1 Defining a Print Form

You define a form by entering the DCL command DEFINE/FORM. This command is restricted to users with OPER privilege. A form definition can include a name, a corresponding number, and optional form characteristics. The DEFINE/FORM command uses the following format:

DEFINE/FORM name number [/qualifiers]

The command in the following example defines the form name CHECKS as the number 3 and defines the margins for the form:

```
$ DEFINE/FORM CHECKS 3/STOCK=DEFAULT -
_$ /MARGIN=(BOTTOM=3,TOP=3,LEFT=6,RIGHT=6)
```

In this example, the form name CHECKS is interpreted to be the same as the form number 3. Note that you cannot abbreviate the form name.

By specifying additional qualifiers with the DEFINE/FORM command, you can define the following characteristics for a particular form:

- A specific image area for the form
- A type of paper stock
- Specific device control modules
- Print job processing attributes

The /STOCK qualifier of the DEFINE/FORM command allows you to associate a type of paper stock with a particular form. This qualifier is useful when you have several forms that have the same paper stock but differ in other ways, such as margin specifications, wrapping, or page dimensions. If you do not specify a stock name with the DEFINE/FORM command, the form name becomes the default value of the /STOCK qualifier.

To delete a defined form, enter the DCL command DELETE/FORM in the following format:

DELETE/FORM form_name

For a detailed description of the DEFINE/FORM command and its qualifiers, see the *VMS DCL Dictionary*.

6.6.2.2 Mounting a Form on a Queue

Mounting a form on a queue associates the stock of a form with an output execution queue. The stock of the form mounted on the output execution queue must match the stock of the default form for the queue, and the stock that is physically mounted on the printer.

You can change the mounted form on a queue in two ways:

- Enter the /FORM_MOUNTED qualifier with the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE.
- Submit a print job with an explicit /FORM=type qualifier in the PRINT command line.

A job submitted without an explicit form specified on the PRINT command line is processed with the default form for the queue. The mounted form on a queue is the form of the last print job submitted with an explicit /FORM=type qualifier in the PRINT command line.

If the form specified in the last PRINT command has a stock type that does not match the stock of the default form on the queue, the job enters a pending state until the stock are made identical. The following command sequence illustrates this point:

```
$ DEFINE/FORM MEMO 6/STOCK=LT_HEAD
$ DEFINE/FORM MANUAL 5/STOCK=STANDARD
$ SET QUEUE/FORM_MOUNTED=MEMO PRINT_QUE
$ PRINT JOB_A.TXT
$ PRINT/FORM=MANUAL JOB_B.TXT
```

In this example, two forms are defined: MEMO and MANUAL. JOB_A.TXT executes on queue PRINT_QUE using the mounted form MEMO. JOB_B.TXT is submitted with the form MANUAL specified on the PRINT command line. JOB_B.TXT enters a pending state because the stock of form MANUAL does not match the stock of form MEMO. A print overrides the current mounted form only if the stocks of the two forms are identical.

6.6.2.3 Establishing a Default Form for a Queue

You create a default form for a specific output execution queue by using the /DEFAULT=FORM=type qualifier option of the DCL command INITIALIZE /QUEUE, START/QUEUE, or SET QUEUE. You must define the form before establishing it as the default form for a queue. Also, the stock of the default form must be identical to the stock of the mounted form or print jobs submitted without an explicit form in the command line will pend until the forms are made identical.

To find out the default form for a particular queue, enter the DCL command SHOW QUEUE/FULL, as shown in the following example:

```
$ SHOW QUEUE/FULL LN01_PRINT
```

```
Printer queue LN01_PRINT, on ALPHA::ALPHA_LCA0, mounted form PORTRAIT (stock=8_5x11)
/BASE_PRIORITY=4 /DEFAULT=(FEED,FLAG,FORM=REPORT(stock=8_5x11),TRAILER=ONE)
/NOENABLE_GENERIC Lowercase /OWNER=[1,4] /PROTECTION=(S:E,O:D,G:R,W:W)
```

In this example, the default form is named REPORT and the stock is named 8_5x11. The stock of the default form matches the stock of the mounted form. All jobs not associated with an explicit form definition in the PRINT command have the form named REPORT by default. Because the stock of the form PORTRAIT matches the stock of the default form REPORT, all jobs submitted to this queue without an explicit form definition are processed.

The default form is associated with a print job at the time the job is executed, unless the job is submitted with an explicit form. Therefore, if you submit a job to a generic queue without specifying a form in the PRINT command line, no form is associated with the job until it is transferred to an execution queue.

6.6.2.4 Changing the Systemwide Default Form

The VMS operating system comes with a systemwide default form named DEFAULT, which corresponds to the form number 0. A queue initialized without a specific /FORM_MOUNTED or /DEFAULT=FORM=type qualifier uses the systemwide default form to process print jobs not explicitly associated with a form definition.

The systemwide DEFAULT form has the following attributes:

```
Form name                    Number      Description

DEFAULT                           0      System-defined default
    /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /TRUNCATE /WIDTH=132
```

The stock of the DEFAULT form is the system-supplied default stock. To change the systemwide default, enter the following command:

```
DEFINE/FORM DEFAULT 0 attribute_qualifier[s]
```

For example, to change the systemwide default form's bottom margin from 6 to 4, and the page length from 66 to 55, enter this command:

```
$ DEFINE/FORM DEFAULT 0 /MARGIN=(BOTTOM=4) /LENGTH=55
```

6.6.3 Specifying Mandatory Queue Attributes

You can control certain print job processing attributes by assigning *job separation pages* to the queue. Unlike default print control features (see in Section 6.6.4), job separation pages cannot be overridden by the PRINT command.

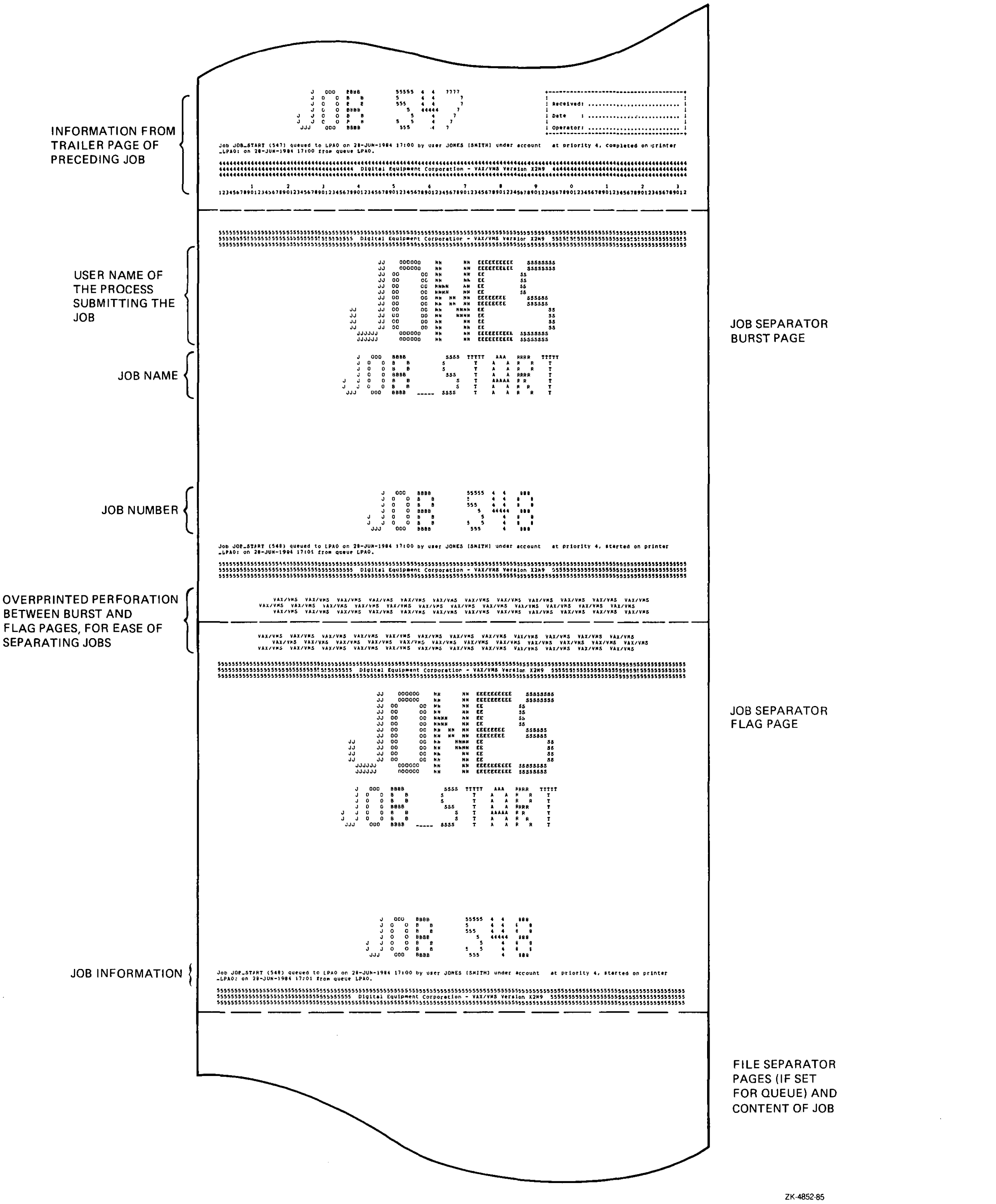
The three mandatory separation pages include job burst, job flag, and job trailer pages. Most sites use only a subset of the available separation pages at a given time. Figure 6–2 shows the output of typical job flag and job burst pages. Figure 6–3 shows the output of a typical trailer page.

You assign job separation pages to a queue by specifying the /SEPARATE qualifier with the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. The /SEPARATE qualifier has the following format:

```
/[NO]SEPARATE=(option[,...])
```

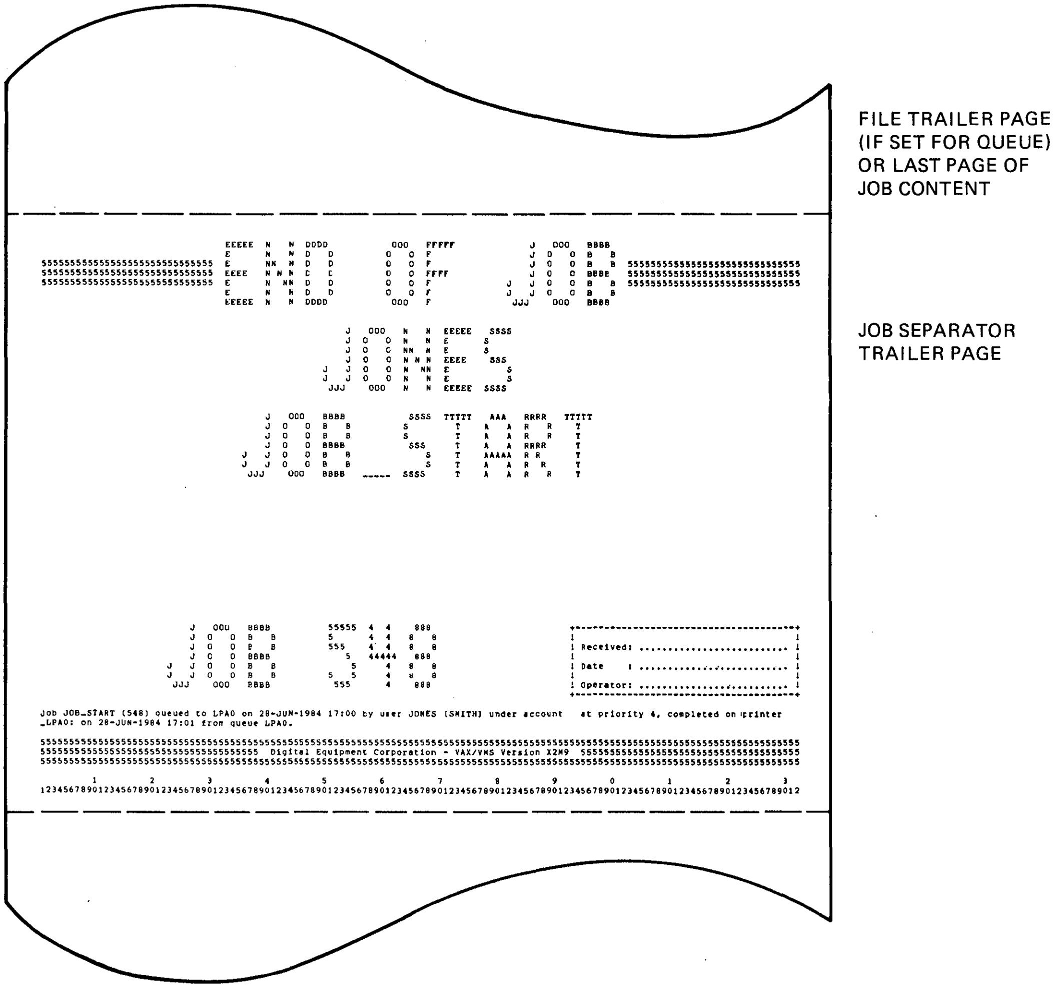
In addition to specifying separation pages, you can also use the /SEPARATE qualifier to specify a list of device control library modules. Typically, device control library modules are used to reset the device to a known state at the end of each job (see Section 6.6.6).

Figure 6–2 Job Flag and Burst Pages


INFORMATION FROM TRAILER PAGE OF PRECEDING JOB
USER NAME OF THE PROCESS SUBMITTING THE JOB
JOB NAME
JOB NUMBER
OVERPRINTED PERFORATION BETWEEN BURST AND FLAG PAGES, FOR EASE OF SEPARATING JOBS
JOB INFORMATION
JOB SEPARATOR BURST PAGE
JOB SEPARATOR FLAG PAGE
FILE SEPARATOR PAGES (IF SET FOR QUEUE) AND CONTENT OF JOB
ZK-4852-85

```
! Received: .............................. !
! Date    : .............../.............. !
! Operator: .............................. !

Job JOB_START (547) queued to LPA0 on 28-JUN-1984 17:00 by user JONES [SMITH] under account   at priority 4, completed on printer
_LPA0: on 28-JUN-1984 17:00 from queue LPA0.

Digital Equipment Corporation - VAX/VMS Version X2N9

          1         2         3         4         5         6         7         8         9         0         1         2         3
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123

Digital Equipment Corporation - VAX/VMS Version X2N9

Job JOB_START (548) queued to LPA0 on 28-JUN-1984 17:00 by user JONES [SMITH] under account   at priority 4, started on printer
_LPA0: on 28-JUN-1984 17:01 from queue LPA0.

Digital Equipment Corporation - VAX/VMS Version X2N9

    VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
    VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
-----------------------------------------------------------------------------------------------------------------
VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
    VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS

Digital Equipment Corporation - VAX/VMS Version X2N9

Job JOB_START (548) queued to LPA0 on 28-JUN-1984 17:00 by user JONES [SMITH] under account   at priority 4, started on printer
_LPA0: on 28-JUN-1984 17:01 from queue LPA0.

Digital Equipment Corporation - VAX/VMS Version X2N9
```

Figure 6–3 Job Trailer Page



```
+--------------------------------------+
!                                      !
! Received: ......................... !
!                                      !
! Date    : ......................... !
!                                      !
! Operator: ......................... !
+--------------------------------------+

Job JOB_START (548) queued to LPA0 on 28-JUN-1984 17:00 by user JONES [SMITH] under account   at priority 4, completed on printer
_LPA0: on 28-JUN-1984 17:01 from queue LPA0.

555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
555555555555555555555555555555555555  Digital Equipment Corporation - VAX/VMS Version X2M9  55555555555555555555555555555555555555
555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555

         1         2         3         4         5         6         7         8         9         0         1         2         3
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

ZK-4850-85

You assign job separation features by specifying one or more of the following options of the /SEPARATE=(option[, . . .]) qualifier:

Option	Description
[NO]FLAG	Specifies that a job flag page is printed at the beginning of each job.
[NO]BURST	Specifies that a job burst page and a job flag page are printed at the beginning of each job.

Option	Description
[NO]RESET=(module[, . . .])	Specifies one or more device control library modules that contain the job reset sequence for the queue. The specified modules from the queue's device control library (by default SYS$LIBRARY:SYSDEVCTL) are used to reset the device each time a job reset occurs. The RESET sequence occurs after any file trailer and before any job trailer. Thus, all job separation pages are printed when the device is in its RESET state. You should assign the RESET job separation feature to queues for printers that support characteristics changeable by the DCL command PRINT/SETUP. The RESET feature ensures that any SETUP features specified for a particular job are not assigned to the next job.
[NO]TRAILER	Specifies that a job trailer page is printed at the end of each job.

By default, no job separation features are assigned to the queue. For more information on the /SEPARATE options, see the *VMS DCL Dictionary*.

Widths greater than 40 characters and less than 200 characters, and lengths of any size greater than 40 characters are supported for file and job separation pages. Pages requested for widths greater than 200 characters are formatted and printed at 200-character widths. Lengths less than 40 characters are extended by that form length until the 40-character threshold is exceeded. Margins are not taken into account when formatting separation pages.

Caution: **All separation pages format information to the width and length of a VMS default form size of 80 characters by 51 lines. Information, therefore, may be truncated, depending on the form sizes you specify. See Section 6.6.5 for information on controlling line and page overflow.**

6.6.3.1 Job Separation Pages

This section gives a detailed description of the characteristics of job separation pages.

Job Flag Page/Burst

The job flag page indicates that a new job follows; the job may include one or more files. Items include the following:

- Header bar: single-segment bar composed of the following elements:
  - Rows of a repeated numeral indicating the sequence of the job in the printer queue.
  - An embedded text string specified by defining the PSM$ANNOUNCE system logical name. The length of the string is limited to one form width. If PSM$ANNOUNCE is not defined, the default text string is "Digital Equipment Corporation" followed by the system version number. ("Digital Equipment Corporation" is abbreviated to "DEC" for shorter form widths.)

- Note: Users may specify a string of up to 255 characters with the /NOTE qualifier of the DCL command PRINT.
- Identification banner for the process submitting the job; includes user name, job name, and job number.
- Job sentence indicating the following information:
  - Job name and number
  - Name of queue to which job was submitted
  - Submission date and time
  - Process username, UIC, and account
  - Job priority
  - Print device name
  - Job start time
  - Execution queue name
- Footer bar: identical to the header bar with the exception that the definition of the system logical name (PSM$ANNOUNCE) is not used as the embedded string. The default text is always used as the embedded string.

Job Trailer Page

The job trailer page indicates the end of a print job; items include the following:

- Header bar: three-segment bar composed of the following elements:
  - Central segment with "END OF JOB" banner
  - Flanking segments with job-sequence numeral
- Identification banner for the process submitting the job; includes job name and job number
- Receipt box with the signoff fields: **Received:**, **Date:**, and **Operator:**
- Job sentence
- Ruler

If you request job burst and file burst pages, a separation burst bar (with the characters VMS) is printed over the page crease between the job flag and job burst pages, and between the file flag and file burst pages, respectively. This facilitates the separation of job and file printouts.

6.6.4 Specifying Default Queue Attributes

To control the default processing attributes of a queue, use the /DEFAULT qualifier of the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. For example, you can specify the default file separation pages. The /DEFAULT qualifier has the following format:

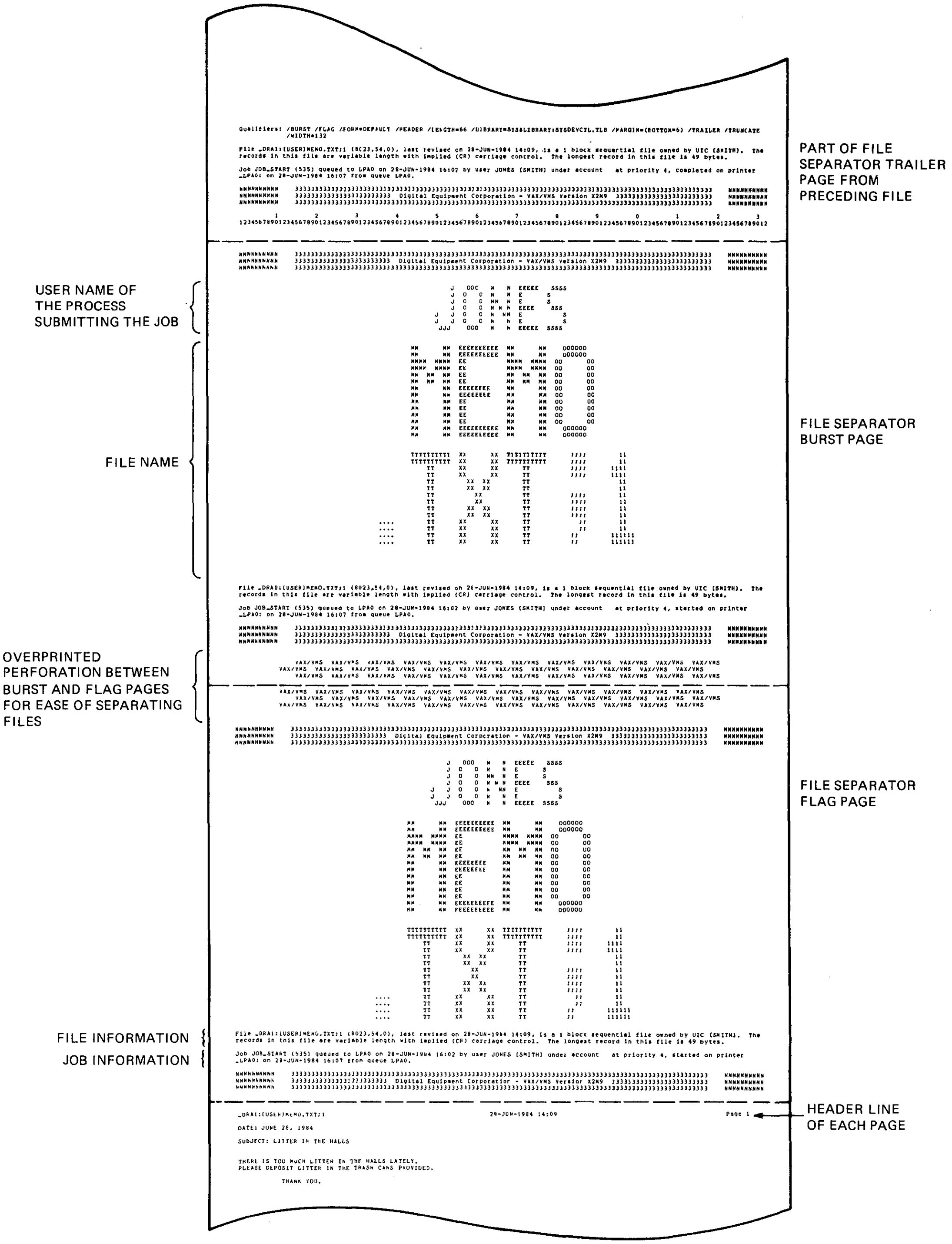
/[NO]DEFAULT=(option[,...])

The following default options are available:

Option	Description
[NO]BURST[=value]	Controls whether a file flag and burst page is printed preceding output. If you specify the value ONE, a flag and burst page is printed once with the first file in the job. If you specify the value ALL, a flag and burst page is printed with each file in the job.
[NO]FEED	Controls whether a form feed is automatically inserted at the end of a page.
[NO]FLAG[=value]	Controls whether a file flag page is printed preceding output. If you specify the value ONE, a flag page is printed once with the first file in the job. If you specify the value ALL, a flag page is printed with each file in the job.
FORM=type	Specifies the default form for a printer, terminal, or server queue. If a job is not submitted with an explicit form definition, then this form will be used to process the job. The systemwide default form (DEFAULT,0) is the default value for this keyword. (See Section 6.6.2.3 for details.)
[NO]TRAILER[=value]	Controls whether a file trailer page is printed following output. If you specify the value ONE, a trailer page is printed once with the last file in the job. If you specify the value ALL, a trailer page is printed with each file in the job.

The value ALL is the default value for options that require a value of ALL or ONE. Figure 6–4 shows the output of typical file flag and file burst pages. Figure 6–5 shows the output of a typical file trailer page.

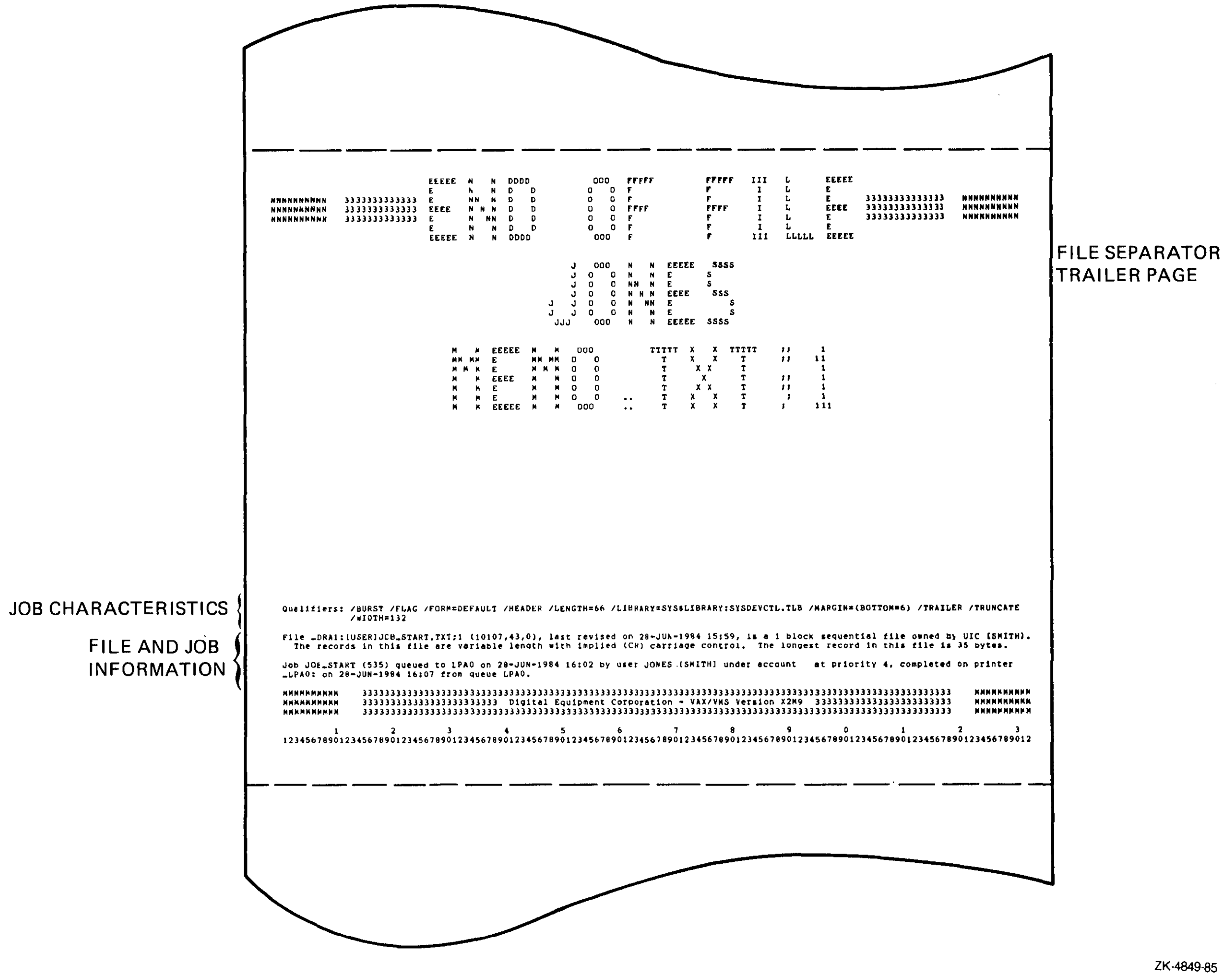
Figure 6–4 File Flag and Burst Pages



```
Qualifiers: /BURST /FLAG /FORM=DEFAULT /HEADER /LENGTH=66 /LIBRARY=SYS$LIBRARY:SYSDEVCTL.TLB /MARGIN=(BOTTOM=6) /TRAILER /TRUNCATE
            /WIDTH=132

File _DRA1:[USER]MEMO.TXT;1 (8023,54,0), last revised on 28-JUN-1984 14:09, is a 1 block sequential file owned by UIC [SMITH].  The
records in this file are variable length with implied (CR) carriage control.  The longest record in this file is 49 bytes.

Job JOB_START (535) queued to LPA0 on 28-JUN-1984 16:02 by user JONES [SMITH] under account   at priority 4, completed on printer
_LPA0: on 28-JUN-1984 16:07 from queue LPA0.

NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    333333333333333333333333  Digital Equipment Corporation - VAX/VMS Version X2M9  333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN

         1         2         3         4         5         6         7         8         9         0         1         2         3
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    333333333333333333333333  Digital Equipment Corporation - VAX/VMS Version X2M9  333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN

                                                 J   OOO   N   N  EEEEE   SSSS
                                                 J  O   O  N   N  E      S
                                                 J  O   O  NN  N  E      S
                                                 J  O   O  N N N  EEEE    SSS
                                              J  J  O   O  N  NN  E          S
                                              J  J  O   O  N   N  E          S
                                               JJJ   OOO   N   N  EEEEE  SSSS

                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO
                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO
                                    MMMM  MMMM  EE          MMMM  MMMM  OO      OO
                                    MMMM  MMMM  EE          MMMM  MMMM  OO      OO
                                    MM  MM  MM  EE          MM  MM  MM  OO      OO
                                    MM  MM  MM  EE          MM  MM  MM  OO      OO
                                    MM      MM  EEEEEEEE    MM      MM  OO      OO
                                    MM      MM  EEEEEEEE    MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO
                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO

                                    TTTTTTTTTT  XX        XX  TTTTTTTTTT      ;;;;      11
                                    TTTTTTTTTT  XX        XX  TTTTTTTTTT      ;;;;      11
                                        TT      XX        XX      TT          ;;;;    1111
                                        TT      XX        XX      TT          ;;;;    1111
                                        TT        XX    XX        TT                    11
                                        TT        XX    XX        TT                    11
                                        TT          XX            TT          ;;;;      11
                                        TT          XX            TT          ;;;;      11
                                        TT        XX    XX        TT          ;;;;      11
                                        TT        XX    XX        TT          ;;;;      11
                              ....      TT      XX        XX      TT            ;;      11
                              ....      TT      XX        XX      TT            ;;      11
                              ....      TT      XX        XX      TT          ;;      111111
                              ....      TT      XX        XX      TT          ;;      111111


File _DRA1:[USER]MEMO.TXT;1 (8023,54,0), last revised on 28-JUN-1984 14:09, is a 1 block sequential file owned by UIC [SMITH].  The
records in this file are variable length with implied (CR) carriage control.  The longest record in this file is 49 bytes.

Job JOB_START (535) queued to LPA0 on 28-JUN-1984 16:02 by user JONES [SMITH] under account   at priority 4, started on printer
_LPA0: on 28-JUN-1984 16:07 from queue LPA0.

NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    333333333333333333333333  Digital Equipment Corporation - VAX/VMS Version X2M9  333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN

                   VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
               VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
                   VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
               VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
                   VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS
               VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS  VAX/VMS


NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    333333333333333333333333  Digital Equipment Corporation - VAX/VMS Version X2M9  333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN

                                                 J   OOO   N   N  EEEEE   SSSS
                                                 J  O   O  N   N  E      S
                                                 J  O   O  NN  N  E      S
                                                 J  O   O  N N N  EEEE    SSS
                                              J  J  O   O  N  NN  E          S
                                              J  J  O   O  N   N  E          S
                                               JJJ   OOO   N   N  EEEEE  SSSS

                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO
                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO
                                    MMMM  MMMM  EE          MMMM  MMMM  OO      OO
                                    MMMM  MMMM  EE          MMMM  MMMM  OO      OO
                                    MM  MM  MM  EE          MM  MM  MM  OO      OO
                                    MM  MM  MM  EE          MM  MM  MM  OO      OO
                                    MM      MM  EEEEEEEE    MM      MM  OO      OO
                                    MM      MM  EEEEEEEE    MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EE          MM      MM  OO      OO
                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO
                                    MM      MM  EEEEEEEEEE  MM      MM    OOOOOO

                                    TTTTTTTTTT  XX        XX  TTTTTTTTTT      ;;;;      11
                                    TTTTTTTTTT  XX        XX  TTTTTTTTTT      ;;;;      11
                                        TT      XX        XX      TT          ;;;;    1111
                                        TT      XX        XX      TT          ;;;;    1111
                                        TT        XX    XX        TT                    11
                                        TT        XX    XX        TT                    11
                                        TT          XX            TT          ;;;;      11
                                        TT          XX            TT          ;;;;      11
                                        TT        XX    XX        TT          ;;;;      11
                                        TT        XX    XX        TT          ;;;;      11
                              ....      TT      XX        XX      TT            ;;      11
                              ....      TT      XX        XX      TT            ;;      11
                              ....      TT      XX        XX      TT          ;;      111111
                              ....      TT      XX        XX      TT          ;;      111111

File _DRA1:[USER]MEMO.TXT;1 (8023,54,0), last revised on 28-JUN-1984 14:09, is a 1 block sequential file owned by UIC [SMITH].  The
records in this file are variable length with implied (CR) carriage control.  The longest record in this file is 49 bytes.

Job JOB_START (535) queued to LPA0 on 28-JUN-1984 16:02 by user JONES [SMITH] under account   at priority 4, started on printer
_LPA0: on 28-JUN-1984 16:07 from queue LPA0.

NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    333333333333333333333333  Digital Equipment Corporation - VAX/VMS Version X2M9  333333333333333333333333    NNNNNNNNNN
NNNNNNNNNN    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    NNNNNNNNNN
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
_DRA1:[USER]MEMO.TXT;1                                       28-JUN-1984 14:09                                              Page 1

DATE: JUNE 28, 1984

SUBJECT: LITTER IN THE HALLS


THERE IS TOO MUCH LITTER IN THE HALLS LATELY.
PLEASE DEPOSIT LITTER IN THE TRASH CANS PROVIDED.

             THANK YOU.
```

PART OF FILE SEPARATOR TRAILER PAGE FROM PRECEDING FILE
USER NAME OF THE PROCESS SUBMITTING THE JOB
FILE NAME
FILE SEPARATOR BURST PAGE
OVERPRINTED PERFORATION BETWEEN BURST AND FLAG PAGES FOR EASE OF SEPARATING FILES
FILE SEPARATOR FLAG PAGE
FILE INFORMATION
JOB INFORMATION
HEADER LINE OF EACH PAGE
ZK-4851-85

Figure 6–5 File Trailer Page



```
                         EEEEE  N   N  DDDD      OOO  FFFFF     FFFFF  III  L      EEEEE
                         E      NN  N  D   D    O   O F         F       I   L      E
NNNNNNNNNN  333333333333 E      NN  N  D   D    O   O F         F       I   L      E      33333333333333  NNNNNNNNNN
NNNNNNNNNN  333333333333 EEEE   N N N  D   D    O   O FFFF      FFFF    I   L      EEEE   33333333333333  NNNNNNNNNN
NNNNNNNNNN  333333333333 E      N  NN  D   D    O   O F         F       I   L      E      33333333333333  NNNNNNNNNN
                         E      N   N  D   D    O   O F         F       I   L      E
                         EEEEE  N   N  DDDD      OOO  F         F      III  LLLLL  EEEEE

                                          J  OOO  N   N EEEEE  SSSS
                                          J O   O N   N E     S
                                          J O   O NN  N E     S
                                          J O   O N N N EEEE   SSS
                                      J   J O   O N  NN E         S
                                      J   J O   O N   N E         S
                                       JJJ   OOO  N   N EEEEE SSSS

                           M   M EEEEE M   M  OOO        TTTTT X   X TTTTT  ;;   1
                           MM MM E     MM MM O   O         T    X X    T    ;;  11
                           M M M E     M M M O   O         T     X     T         1
                           M   M EEEE  M   M O   O         T     X     T    ;;   1
                           M   M E     M   M O   O         T    X X    T    ;;   1
                           M   M E     M   M O   O  ..     T   X   X   T     ;   1
                           M   M EEEEE M   M  OOO   ..     T   X   X   T     ;  111

Qualifiers: /BURST /FLAG /FORM=DEFAULT /HEADER /LENGTH=66 /LIBRARY=SYS$LIBRARY:SYSDEVCTL.TLB /MARGIN=(BOTTOM=6) /TRAILER /TRUNCATE
            /WIDTH=132

File _DRA1:[USER]JCB_START.TXT;1 (10107,43,0), last revised on 28-JUN-1984 15:59, is a 1 block sequential file owned by UIC [SMITH].
  The records in this file are variable length with implied (CR) carriage control.  The longest record in this file is 35 bytes.

Job JOE_START (535) queued to LPA0 on 28-JUN-1984 16:02 by user JONES [SMITH] under account   at priority 4, completed on printer
_LPA0: on 28-JUN-1984 16:07 from queue LPA0.

MMMMMMMMMM    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    MMMMMMMMMM
MMMMMMMMMM    333333333333333333333333  Digital Equipment Corporation - VAX/VMS Version X2M9  333333333333333333333333    MMMMMMMMMM
MMMMMMMMMM    33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333    MMMMMMMMMM

         1         2         3         4         5         6         7         8         9         0         1         2         3
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

6.6.4.1 File Separation Pages

This section gives a detailed description of the characteristics of file separation pages.

File Flag/Burst Pages

The file flag and burst pages indicate that a print file follows. The file may include one or more of the following items:

- Header bar: three-segment bar composed of the following elements:
  - A central segment identical to the job header bar
  - Flanking segments with rows of a repeated character indicating the sequence of the file in the job
- Note: Users may specify a string up to 255 characters in length using the /NOTE qualifier of the DCL command PRINT

- Identification banner for the process submitting the job: user name, file name
- File sentence: indicates the following information:
  - Full file specification, including device and directory, file name, type, version, and revision time and date
  - File size in blocks
  - File organization
  - File owner's UIC
  - File record characteristics: record type, carriage control, size of longest record
- Job sentence
- Footer bar: identical to header bar except that the embedded text string is "Digital Equipment Corporation - Version Vn.n"

File Trailer Page

The file trailer page indicates the end of a print file. Items include the following:

- Header bar: five-segment bar composed of the following elements:
  - Central segment with "END OF FILE" banner
  - Inner flanking segments with job-sequence numeral
  - Outer flanking segments with file-sequence character
- Identification banner for the process submitting the job; includes user name, file name
- Qualifier phrase: indicates the print, queue, and form qualifiers active when the job was submitted; nonactive qualifiers (except /NORECORD_BLOCKING and /NOFEED) are not included
- File sentence
- Job sentence
- Footer bar identical to file flag/file burst footer bar
- Ruler: a sequence of numbers counting to the end of the form

6.6.5 Controlling Page and Line Overflow

DIGITAL recommends that you control line and page overflow by using form definitions. To do this, you must set terminals and printers to avoid wrapping or truncating the print line at the physical limits of the device.

Note: **The device driver operates on settings for the /TRUNCATE and /WRAP qualifiers specified with the DCL commands SET PRINTER and SET TERMINAL only *after* the print symbiont has finished formatting data.**

Different forms may have different right, left, top, and bottom margin settings. By using forms to control page and line overflow, users can switch from one form to another without requiring operators to stop the queue, alter the device setup, and restart the queue. Forms with the same value for the /STOCK qualifier are automatically mountable without operator assistance.

You control line overflow by using the following qualifiers to the DCL command DEFINE/FORM:

Qualifier	Function
/[NO]TRUNCATE	Specifies that data in the right margin be discarded. The default is /TRUNCATE. If you specify /TRUNCATE, you cannot specify /WRAP; the /TRUNCATE qualifier forces /NOWRAP.
/[NO]WRAP	Specifies that a carriage return and line feed be inserted when the right margin is exceeded. The default is /NOWRAP. If you specify /WRAP, you cannot specify /NOTRUNCATE.

Page overflow errors are handled by the /DEFAULT=FEED option of the following commands:

```
INITIALIZE/QUEUE
START/QUEUE
SET QUEUE
PRINT/[NO]FEED
```

This qualifier controls whether a formfeed character is automatically inserted when the symbiont detects entry into the bottom margin area.

You use the /PASSALL qualifier with the PRINT command to bypass all formatting performed by the print symbiont. The default is /NOPASSALL. Use this qualifier in cases where the print symbiont formatting may interfere with the desired formatting of the file. The /PASSALL qualifier sends I/O to the device driver with all formatting suppressed, and behaves as follows:

- Does not interpret FORTRAN or print carriage control characters
- Does not perform line or page overflow error handling
- Does not interpret escape sequences

6.6.6 Using Device Control Libraries

A device control library is a text library that contains user-written modules for setting up printers to perform special printer functions. Device control library modules can be used for the following purposes:

- To insert device-dependent escape sequences that set up programmable printers for selected print options such as point size, character set, and bold or italic print
- To insert text at specific points in th^e processing of a print job such as at the beginning of a file, the beginning of each page, or at the end of each job. (Modules designed for this purpose can be used on both programmable and nonprogrammable printers.)

Use the following DCL commands to set up device control library modules for processing print jobs:

Command	Description
DEFINE/FORM/SETUP	Specifies one or more modules that set up the device when the form is mounted before each file.
DEFINE/FORM/[NO]PAGE_SETUP	Specifies one or more modules that set up the device before each page.
INITIALIZE/QUEUE/LIBRARY START/QUEUE/LIBRARY	Specifies the file name of the device control library. The default library is SYS$LIBRARY:SYSDEVCTL.TLB. This parameter must contain a file name only, without an extension or a directory.
INITIALIZE/QUEUE/SEPARATE=[NO]RESET START/QUEUE/SEPARATE=[NO]RESET SET QUEUE/SEPARATE=[NO]RESET	Specifies one or more modules to be extracted from the device control library and copied to the device at the end of each job. This module can be used to restore the device to a known state at the end of each job. The default is NORESET.
PRINT/SETUP	Calls for the specified modules to be extracted from the device control library and copied to the printer before a file is printed.

To use a device control library module, perform the following steps:

1. Create a device control library by entering the following command:

   ```
   $ LIBRARY/CREATE/TEXT SYS$LIBRARY:SYSDEVCTL.TLB
   ```

2. Determine the contents of the module—either the text to be inserted, or the escape sequences needed for the desired printer setup. To determine the proper escape sequences for a printer option, refer to the operation guide for the specific printer.
3. Create a module file and enter the appropriate escape sequences, carriage control characters, or text. You create and edit a module file as you would any other text file.
4. Insert the module into the device control library by entering the following command:

   LIBRARY/INSERT/TEXT library-file module-file

5. Assign the device control library to a queue by entering the following command:

```
INITIALIZE/QUEUE/LIBRARY=filename  queue-name
```

Note that this step is not necessary if you use the default library SYSDEVCTL.TLB.

6 Request a device control library module by entering either the PRINT /FORM or PRINT/SETUP command, for example:

```
$ PRINT/SETUP=MODULE1 TEST1.TXT/QUEUE=PDQ_QUE
```

(See the *VMS Librarian Utility Manual* for more information on creating libraries and inserting modules.)

6.6.6.1 Assigning a Library to an Output Queue

You assign a device control library to an output queue by specifying the /LIBRARY qualifier with the INITIALIZE/QUEUE or START/QUEUE command in the following format:

```
INITIALIZE/QUEUE/START/LIBRARY=filename  queue-name
```

The file name is the name of the library file that contains the desired modules. Libraries must be in SYS$LIBRARY and have the default file type TLB. If you do not specify a file name, by default the file name is SYS$LIBRARY:SYSDEVCTL.TLB. You can use the /LIBRARY qualifier to specify an alternate device control library, for example:

```
$ INITIALIZE/QUEUE/LIBRARY=MYDEVCTL PDQ_QUE
```

Note: **If you specify a file name in this parameter, do NOT include the directory, file extension, or version number. The system assumes that the file is in SYS$LIBRARY and has the extension TLB. If you copy a library file from another node, be sure that the new library has a unique file name.**

Operations that request a particular device control library module use the module from the library specified for the queue. If you have a small configuration of printers, and normally use only a few modules, you usually store all modules in a single library and assign that same library to each printer queue.

For sites with a large number of different printers, you usually create and assign a separate device control library for each printer queue. By using separate libraries for each printer queue, you can create modules that have the same name and perform the same functions, but contain escape sequences unique to the specific type of printer. If you use a single library to store modules for different types of printers, make sure that each module has a unique name.

You use the following command format to display a listing of all modules contained within a specified library:

```
$ LIBRARY/LIST/FULL SYS$LIBRARY:library-name.TLB
```

Note: **To add or delete a module from a library, you must stop all output queues assigned to that library.**

6.6.6.2 Requesting a Library Module

The system manager can control the printer setups available to users by defining forms that are associated with one or more device control library modules for a particular queue. The DCL command DEFINE/FORM/SETUP associates a form with a device control library module that sets up the printer before each file of a job is printed. The command DEFINE/FORM/PAGE_SETUP associates a form with a module that sets up the printer before each page is printed.

By entering the appropriate PRINT/FORM or PRINT/SETUP command, users can request a device control module to set up a specific printer option or to insert text at the beginning of each file or at the beginning of each page.

The PRINT/FORM command has certain advantages over the PRINT/SETUP command. The form name specified with the PRINT/FORM command is checked for validity when the command is entered. However, the module names are not checked until the file is printed, because the name of the device control library is not determined until that time. If a user makes an error while entering the module name in the PRINT/SETUP command, the job may not print. To reduce the chance of user errors occurring when the PRINT/SETUP command is entered, DIGITAL recommends that system managers use the DEFINE/FORM/SETUP command to assign a library module to a specified form. Once a form is properly defined, then use the PRINT/FORM command to set up the printer for the specified printer function.

Caution: **If you enter an erroneous module name when entering the PRINT/SETUP command, the print symbiont writes an error message to the output device stating that the module does not exist. The print job does not print and may be lost. By default, jobs are NOT retained in an output queue. (See Section 6.4.3 for information on retaining jobs.)**

To reset the printer device to a known state at the end of each job, you use the /SEPARATE=RESET qualifier when the queue is initialized, for example:

```
$ INITIALIZE/QUEUE/LIBRARY=MYDEVCTL/SEPARATE=RESET=MODULE2 PDQ_QUE
```

The RESET keyword sets up the printer according to the reset sequence contained in the specified device control module. You can use the /SEPARATE=RESET qualifier with the INITIALIZE/QUEUE, START /QUEUE, or SET QUEUE command. In addition to outputting the RESET sequence at the end of each job, the RESET modules are also output when the queue is started. This ensures that the first job prints correctly.

See Example 6–1 for a sample session in which device control library modules are used to process a print job. In this session, two device control modules are created and inserted into the library file MYDEVCTL.TLB, a print job (REPORT.TXT) is processed using the MODULE1 setup, and at the end of the job, the printer is reset to the MODULE2 setup.

Example 6–1 Using Device Control Library Modules

```
$ LIBRARY/CREATE/TEXT SYS$LIBRARY:MYDEVCTL.TLB
$ CREATE MODULE1.TXT

    !enter printer escape sequences or text for module1
    !type CTRL/Z to EXIT

$ CREATE MODULE2.TXT

    !enter printer escape sequences or text for module2
    !type CTRL/Z to EXIT

$ LIBRARY/INSERT SYS$LIBRARY:MYDEVCLT.TLB/TEXT MODULE1
$ LIBRARY/INSERT SYS$LIBRARY:MYDEVCLT.TLB/TEXT MODULE2
$ INITIALIZE/QUEUE/START/ON=TTA9:/LIBRARY=MYDEVCTL PDQ_QUE
$ SET QUEUE/SEPARATE=RESET=MODULE2 PDQ_QUE
$ SHOW QUEUE/FULL PDQ_QUE

Terminal queue PDQ_QUE, on TOAD::TTA9
   /BASE_PRIORITY=4 /DEFAULT=(FEED) /FORM=DEFAULT /LIBRARY=MYDEVCTL
   /OWNER=[1,4] /PROTECTION=(S:E,O:D,G:R,W:W) /SEPARATE=RESET=(MODULE2)

$ DEFINE/FORM/SETUP=MODULE1 FORM1 1
$ PRINT/FORM=FORM1 REPORT.TXT/QUEUE=PDQ_QUE
Job REPORT (Queue PDQ_QUE, entry 619) started on PDQ_QUE
$
```

6.6.7 Assigning a Logical Queue

To assign a logical queue to a specific printer, terminal, or server queue, use the DCL command ASSIGN/QUEUE. This command is restricted to users with OPER privilege or E access to the queues. Jobs sent to a logical queue are held until the logical queue is assigned to an output execution queue and both queues are started. A logical queue can be used to hold print jobs of low-priority users for printing during off-peak hours.

When inserted in a command procedure, the following commands initialize and start the printer queue MAYA_LPA0 and initialize the logical queue named HOLD_QUE:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  MAYA_LPA0
$ INITIALIZE/QUEUE  HOLD_QUE
```

Jobs sent to the printer queue MAYA_LPA0 are queued for printing. However, any jobs sent to the logical queue HOLD_QUE are held until HOLD_QUE is assigned to MAYA_LPA0 and started (usually from the DCL command level) as follows:

```
$ ASSIGN/QUEUE  MAYA_LPA0  HOLD_QUE
$ START/QUEUE  HOLD_QUE
```

To deassign a logical queue, use the following procedure:

1. Stop the logical queue by specifying the STOP/QUEUE/NEXT command.
2. Deassign the logical queue by specifying the DCL command DEASSIGN /QUEUE.

The following commands deassign the logical queue HOLD_QUE:

```
$ STOP/QUEUE/NEXT  HOLD_QUE
$ DEASSIGN/QUEUE  HOLD_QUE
```

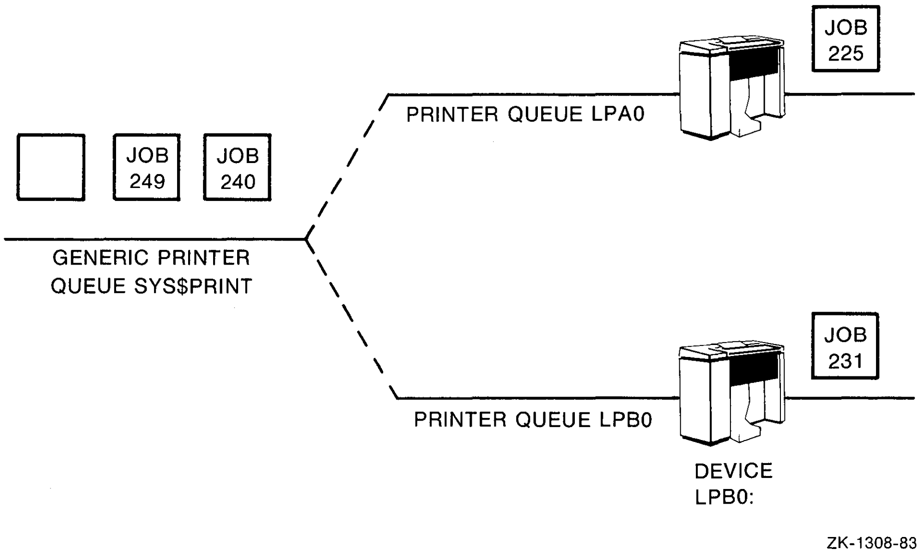
The *VMS DCL Dictionary* describes the DEASSIGN/QUEUE command in detail.

6.6.8 Setting Up Generic Output Queues

Generic output queues distribute the processing of print jobs among print devices that are of the same type. For example, if your system has more than one line printer of the same type, you should create at least one generic output queue. Because print jobs submitted with the PRINT command are sent to the queue SYS$PRINT by default, you should establish SYS$PRINT as a generic queue, assigning to it printers that perform basic print operations.

When print jobs are sent to a generic output queue, they are held there until one of the assigned printer or terminal queues becomes available. Figure 6–6 gives a conceptual view of a generic output queue.

Figure 6–6 Generic Output Queue



You establish a generic queue by specifying the /GENERIC qualifier with either the INITIALIZE/QUEUE or START/QUEUE command. See the *VMS DCL Dictionary* for a complete description of how the /GENERIC qualifier is used with these commands.

The two methods of using the /GENERIC qualifier to define the output execution queues associated with a generic queue are as follows:

- **Using the /[NO]ENABLE_GENERIC qualifier**—One method is to NOT specify a queue name as an argument to the /GENERIC qualifier. Instead, you control which queues have generic printing enabled by using the /[NO]ENABLE_GENERIC qualifier in the command line that initializes and starts each output execution queue. The default is /ENABLE_GENERIC; therefore, any output execution queue that has generic printing enabled and is of the same type (either printer, terminal, or server) will accept jobs from the generic output queue. Those queues initialized and started with the /NOENABLE_GENERIC qualifier will NOT accept jobs from the generic output queue.

If you enable generic processing on a server execution queue, using the method just described, the system checks to see if the user-written symbiont specified with the /PROCESSOR qualifier is the same for both the generic server queue and the execution server queue.

- **Explicitly naming the execution queue(s)**—A second method is to explicitly specify a name or a list of names of target output execution queues (either printer, terminal or server) as arguments to the /GENERIC qualifier. Usually, when you create a generic output queue, you specify a list of type-specific target queues. In this way, the generic output queue directs jobs to a single type of output execution queue. Thus, you can control whether the jobs submitted to a generic output queue are output on a line printer or a terminal printer, or are sent to a user-written symbiont for processing.

Both methods of assigning output execution queues to a generic queue are described in detail in the following sections.

Note: **The /DEFAULT, /FORM_MOUNTED, and /SEPARATE qualifiers are not meaningful for generic output queues. Consequently, the DCL commands INITIALIZE/QUEUE and START/QUEUE reject these qualifiers if /GENERIC is also specified. If the attributes of an established generic output queue are modified, these three qualifiers may be accepted. However, they have no effect on the operation of the generic queue.**

6.6.8.1 Using the /[NO]ENABLE_GENERIC Qualifier

On systems with a small number of printers, you should use the /[NO]ENABLE_GENERIC qualifier with the INITIALIZE/QUEUE/START command to define which output execution queues are assigned to the generic output queue. All output execution queues that are not explicitly disabled for generic printing (by using the /NOENABLE_GENERIC qualifier) can accept jobs from the generic output queue.

Output execution queues for line printers that are in remote locations, that use special forms, or that possess unique printer characteristics should be set up with the /NOENABLE_GENERIC qualifier, to exclude them from accepting jobs from a generic output queue.

The following command procedure example creates four different printer queues, including a generic output queue:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPAO: QUEUE_A
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPBO: QUEUE_B
$ INITIALIZE/QUEUE/START/NOENABLE_GENERIC/ON=LPCO: QUEUE_C
$ INITIALIZE/QUEUE/START/GENERIC  SYS$PRINT
```

In this procedure, because no output execution queue names were specified with the /GENERIC qualifier, the queues that have generic printing enabled by default, QUEUE_A and QUEUE_B, are assigned to the generic queue SYS$PRINT. The queue QUEUE_C is not assigned to the generic output queue because generic printing is explicitly disabled.

On systems with only a few printers of the same type, using the /[NO]ENABLE_GENERIC qualifier method is adequate. On larger systems with many different types of printers, this method is not recommended.

Using the /[NO]ENABLE_GENERIC qualifier method allows you to assign only one type of output execution queue to the generic queue. You use the /DEVICE qualifier to specify the type of output execution queue assigned to the generic queue (specifying either printer, terminal, or server). To set up a generic output queue that serves both printer and terminal queues, you must use the explicit assignment method described in the next section.

6.6.8.2 Assigning Generic Output Queues Explicitly

On systems with a large number of printers of many different types, you should explicitly assign printer queues to a generic output queue by entering a queue name or list of names as the argument to the /GENERIC qualifier. The following command procedure example creates four different printer queues, including a generic output queue:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPA0: Q_A
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPB0: Q_B
$ INITIALIZE/QUEUE/START/DEFAULT=(FLAG,TRAILER,NOFEED)/ON=LPC0:  Q_C
$ INITIALIZE/QUEUE/START/GENERIC=(Q_A,Q_B)  SYS$PRINT
```

In this procedure, the printer queues Q_A and Q_B are assigned to the generic output queue SYS$PRINT. The printer assigned to Q_C uses special forms and is not suited for general printing. Therefore, it is not assigned to the generic output queue SYS$PRINT. Print files queued by default to SYS$PRINT are assigned to Q_A or Q_B, depending on which printer is available.

By specifying a list of queue names with the /GENERIC qualifier, you can assign printer, terminal, or server queues to a generic output queue. This method also allows you to create more than one generic output queue. For example, you may want to create a separate generic output queue for each set of printers having similar characteristics:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPA0: Q_LPA
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPB0: Q_LPB
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPC0: Q_LPC
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=TTD5: Q_TTD5
$ INITIALIZE/QUEUE/START/GENERIC=(Q_LPA,Q_LPB)  SYS$PRINT
$ INITIALIZE/QUEUE/START/GENERIC=(Q_LPC,Q_TTD5)  SMALLFORM
```

This command procedure creates two generic output queues, SYS$PRINT for routine print jobs, and SMALLFORM for print jobs requiring special size forms. Here it is assumed that printer queues Q_LPC and Q_TTD5 are set up for special form jobs.

6.6.9 Setting Up Spooled Devices

Spooling allows application programs to write output to an intermediate storage device so that the printer targetted to print that output remains available to other system users while the program is running.

Spooling is usually used on time shared systems that run applications that would place a high demand on a printer. Application programs often demand input and produce output at irregular intervals during their execution. If spooled devices are set up, the file system intercepts the data storage device (such as a disk) until the application is finished executing and the file is closed. The system then submits the output file to the printer queue associated with the spooled device for immediate execution.

To establish a printer as a spooled device, use the DCL command SET DEVICE/SPOOLED in the following format:

```
$ SET DEVICE/SPOOLED[=(queue-name[:],intermediate-disk-name[:])]
```

This command is restricted to users with the OPER privilege. The *VMS DCL Dictionary* describes the SET DEVICE command in detail. For each output spooled device, you can use the SET DEVICE command to specify both the printer queue to which spooled files are queued and an intermediate device to which spooled files are written. By default, the queue name is the same as the printer device name. The intermediate device name is the translation of the logical name SYS$DISK. The name is translated when you enter the command SET DEVICE/SPOOLED.

You should always specify the intermediate disk and queue explicitly. Whenever a file is copied to a spooled device, it is queued for printing on the queue specified by the SET DEVICE command when the file is closed. If you assign a spooled device to a generic queue, a job output to that device is sent to the generic queue, which in turn places the job in one of its assigned queues. As a result, a job copied to LPA0, for example, may not necessarily print on the printer LPA0, but instead may print on one of the other devices assigned to the generic queue.

When you select an intermediate device, ensure that it has sufficient free space for the volume of queued output. If you plan to enforce disk quotas on the intermediate device, ensure that all expected users of the spooled device have a quota authorized on the intermediate device.

The following example gives the steps for setting up a typical spooled device configuration for a system with one line printer:

```
$ SET PRINTER/LOWER  LPA0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPA0: SYS$PRINT
```

(See Section 6.6.10.1 for more information on setting up a single-printer configuration.)

After establishing a device as spooled, you should test the device, because errors in disk or queue names are not detected until spooling is attempted. To test a spooled device, use a command procedure similar to the following:

```
!         *****TESTING SPOOLED DEVICE***
$ SET DEVICE/SPOOLED=(SYS$PRINT,SYS$SYSDEVICE:) LPA0:
$ CREATE TEST.LIS
   !Add the first test record here.
   !CTRL/Z to exit the file
$ COPY TEST.LIS LPA0:
$ EXIT
```

To set up spooled devices, enter the appropriate commands in your site-specific command procedure (SYS$MANAGER:SYSTARTUP_V5.COM). You can, as necessary, use the SET DEVICE command with the /NOSPOOLING qualifier to disable spooling to spooled printers and terminals; however, you must stop the corresponding queues before you can change the spooling status.

6.6.10 Guidelines for Setting Up Queues for Spooled Devices

The examples in this section apply to print devices in a single-node environment. In a cluster environment, clusterwide queues are established to control print job processing. The *VMS VAXcluster Manual* provides a detailed description of how to set up queues for print devices in a clustered environment.

Use the following guidelines to set up printer queues for spooled devices:

- Specify the proper characteristics for each printer with the DCL command SET PRINTER. (The *VMS DCL Dictionary* describes the SET PRINTER command in detail.)
- Control line overflow with the /TRUNCATE and /WRAP qualifiers of the DCL command DEFINE/FORM. The form controls the line and page overflow, therefore, set the printer or terminal width and length to the physical limits of the device. (See Section 6.6.5 for more information on these qualifiers.)
- Spool the device with the DCL command SET DEVICE/SPOOLED.
- To produce output on a spooled line printer, initialize a printer queue with the same name as the spooled printer and start that queue. You can also assign a queue a name that is different from the printer's physical device name by using the /ON qualifier with the INITIALIZE command (see Section 6.3.2).
- Assign one printer queue the name SYS$PRINT. For systems with one printer, assign that printer's queue the name SYS$PRINT. If more than one line printer is on the system, make at least one printer queue a generic queue and assign it the name SYS$PRINT. Section 6.6.8 describes how to establish generic output queues.

The next sections illustrate some of the most common configurations of output execution queues and spooled line printers.

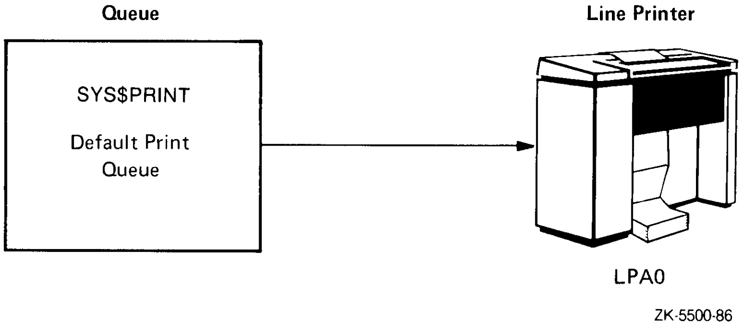
6.6.10.1 Single-Printer Configuration

Figure 6–7 illustrates a typical queue configuration for a system with one line printer. Enter the following commands in your site-specific SYSTARTUP_V5.COM procedure to produce the results listed below:

```
$ SET PRINTER/LOWER  LPA0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPA0: SYS$PRINT
```

1. The characteristics of LPA0 are set.
2. The line printer LPA0 is set spooled.
3. A printer queue for LPA0 is initialized, started, and assigned the name SYS$PRINT.
4. All print jobs are placed by default in a queue named SYS$PRINT and are printed from that queue.

Figure 6–7 Setting Up a Printer Queue on a System with One Printer



6.6.10.2 Two-Printer Configuration

Figure 6–8 illustrates a typical spooling and queuing configuration for a system with two line printers that have the same characteristics. The following commands produce the results listed below:

```
$ !
$ ! Set printer characteristics, set LPA0 printer spooled,
$ ! and set up queue PRINT_A.
$ !
$ SET PRINTER/LOWER LPA0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPA0: PRINT_A
$ !
$ ! Set printer characteristics, set LPB0 spooled,
$ ! and set up queue PRINT_B.
$ !
$ SET PRINTER/LOWER  LPB0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPB0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPB0: PRINT_B
$ !
$ ! Set up the generic queue SYS$PRINT
$ !
$ INITIALIZE/QUEUE/START/GENERIC SYS$PRINT
```

1 The characteristics of device LPA0 are set.

2 The line printer LPA0 is set spooled.

3 The printer queue PRINT_A is initialized and started.

4 The characteristics of device LPB0 are set.

5 The line printer LPB0 is set spooled.

6 The printer queue PRINT_B is initialized and started.

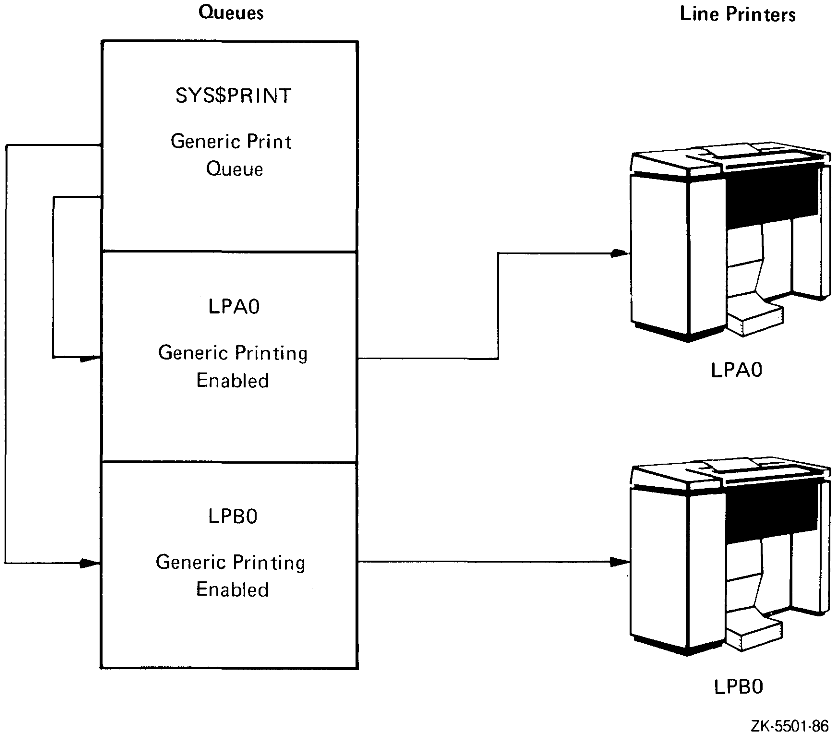
7 The generic queue SYS$PRINT is initialized and started and assigned the printer queues PRINT_A and PRINT_B by default (see Section 6.6.8.1).

8 Print jobs sent with an explicit queue name entered in the PRINT command line are placed in the queue for the specific printer.

9 Print jobs sent without specifying a queue name in the PRINT command are placed by default in the generic queue SYS$PRINT. Jobs are then scheduled for processing in PRINT_A or PRINT_B, depending on which print device is available.

10 Spooled print files destined for either device LPA0 or LPB0 are placed in the generic queue SYS$PRINT, which is associated with both printers. From the generic queue, these jobs are printed on whichever printer is available. (Alternatively, you could set the devices spooled and assign each device the queue name that is associated with that device. This ensures that jobs print on the same printer to which the file was spooled.)

Figure 6–8 Setting Up Printer Queues on a System with Two Printers



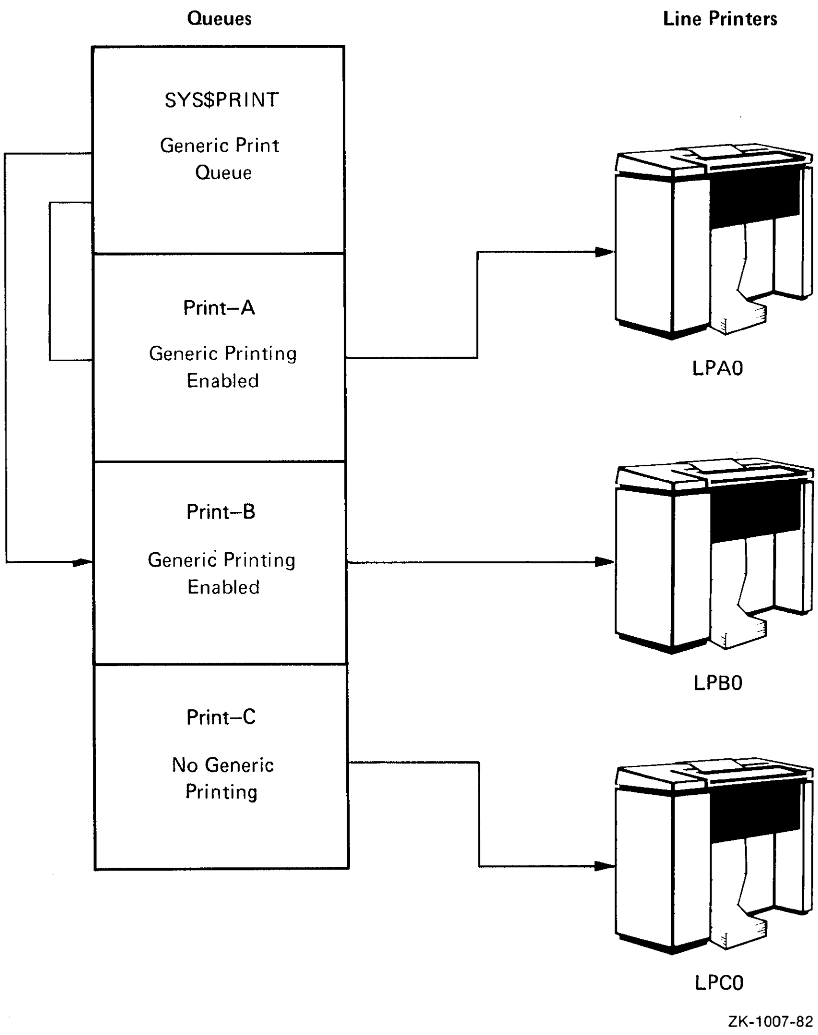
6.6.10.3 Three-Printer Configuration

Figure 6–9 illustrates a typical spooling and queuing configuration for a system with three line printers: two that have the same characteristics and one that uses special forms, has unique printer characteristics, or is in a remote location. The configuration shown in Figure 6–9 is similar to the one in Figure 6–8, except that the spooled printer LPC0 has been added and queue PRINT_C has been initialized and started. Because of the special forms, unique characteristics, or the remote location, printer LPC0 is not suited for general printing. Thus, the following commands produce the same results as those listed in Figure 6–8, with noted exceptions:

```
$ !
$ ! Set printer characteristics, set LPA0 spooled,
$ ! and set up queue PRINT_A.
$ !
$ SET PRINTER/LOWER  LPA0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPA0:
$ INITIALIZE/QUEUE/DEFAULT=FLAG/START/ON=LPA0: PRINT_A
$ !
$ ! Set printer characteristics, set LPB0 spooled,
$ ! and set up queue PRINT_B.
$ !
$ SET PRINTER/LOWER  LPB0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPB0:
$ INITIALIZE/QUEUE/DEFAULT=FLAG/START/ON=LPB0: PRINT_B
$ !
$ ! Set printer characteristics, set LPC0 spooled,
$ ! and set up queue PRINT_C with generic queuing disabled.
$ !
$ SET PRINTER/LOWER/PAGE=32  LPC0:
$ SET DEVICE/SPOOLED=LPC0  LPC0:
$ INITIALIZE/QUEUE/DEFAULT=FLAG/NOENABLE_GENERIC/START/ON=LPC0: PRINT_C
$ !
$ ! Set up the generic printer queue SYS$PRINT
$ !
$ INITIALIZE/QUEUE/GENERIC/START  SYS$PRINT
```

1. The characteristics for LPC0 are set.
2. The line printer LPC0 is set spooled.
3. The printer queue PRINT_C is initialized and started with generic printing disabled.
4. Only print jobs explicitly directed to the printer LPC0 or queue PRINT_C are placed in PRINT_C. No print jobs are directed to PRINT_C from the generic queue SYS$PRINT.

Figure 6–9 Setting Up Printer Queues on a System with Three Printers



6.6.10.4 Configuration for Remote Printers on a Terminal Server

Figure 6–10 illustrates a typical spooling and queuing configuration for a VMS operating system that is part of a LAT network. The VMS system connects by way of the LAT protocol with a terminal server that supports two printers.

The following commands produce the results listed in Figure 6–10:

```
$ !
$ ! Set up local characteristics for the devices.
$
$   SET TERMINAL LTA1: /PERM /DEVICE=LN03 /WIDTH=255 /PAGE=60 -
                /LOWERCASE /NOBROAD
$   SET TERMINAL LTA2: /PERM /DEVICE=LA210 /WIDTH=255 /PAGE=66 -
                /NOBROAD
$
$ ! Define a form to use with the remote printers.
$
$   DEFINE/FORM LN_FORM 10 /WIDTH=80 /STOCK=DEFAULT /TRUNCATE
$
$ ! Set up the printers as spooled devices.
$
$   SET DEVICE LTA1: /SPOOLED=(LN03$PRINT,SYS$SYSDEVICE:)
$   SET DEVICE LTA2: /SPOOLED=(LA210$PRINT,SYS$SYSDEVICE:)
$
$ ! Initialize and start the remote printer queues.
$
$   INITIALIZE/QUEUE /START /PROCESSOR=LATSYM /FORM_MOUNTED=LN_FORM -
             /RETAIN=ERROR /DEFAULT=(NOBURST,FLAG=ONE,NOTRAILER) -
             /RECORD_BLOCKING LN03$PRINT /ON=LTA1:
$   INITIALIZE/QUEUE /START /PROCESSOR=LATSYM /RETAIN=ERROR -
             /DEFAULT=(NOBURST,FLAG=ONE,NOTRAILER) /RECORD_BLOCKING -
              LA210$PRINT /ON=LTA2:
$
```

1. The local characteristics of LTA1 and LTA2 are set. The local characteristics determine how the VMS service node packages data to send across the Ethernet to the terminal server. Include the /NOBROADCAST qualifier because printers are non-interactive devices.

   Note that you must establish the VMS logical devices, LTA1 and LTA2, in the LAT startup command procedure (LTLOAD.COM) before you can assign characteristics to them in this command. See *Guide to Setting Up a VMS System* for a description of editing LTLOAD.COM.

2. Define a form, if necessary, making sure to use a form number that has not already been used. This protects against page overflow.

3. Set the printers LTA1: and LTA2: spooled.

4. Initialize and start the queues LN03$PRINT and LA210$PRINT. (The queue manager must be started before executing this line.)

5. Use LATSYM, the LAT symbiont process, to deliver data from the VMS service node to the printer. The /RETAIN=ERROR qualifier specifies that any jobs that complete unsuccessfully remain in the queue.

Figure 6–10 Configuration for Remote Printers on a Terminal Server

```
  VMS                                     TERMINAL       REMOTE
 QUEUES              ETHERNET              SERVER        PRINTERS
                                            NODE

LN03$PRINT
 |                                                      ----------
 |                                       -------        |        |
 --> LTA1: . .                           |     |        | LTA1:  |
             .                           |     |------|        |
               .                         |     |        |        |
LA210$PRINT     .........................|     |        ----------
 |              .                        |     |        ----------
 |                .                      |     |------|        |
 --> LTA2: . .                           |     |        | LTA2:  |
                                         -------        |        |
                                                        |        |
                                                        ----------
```

6.7 Managing the Card Reader

Information in this section applies to the DIGITAL CR-11 card reader, which is used to read computer card decks. Users may submit the two following types of card decks for processing:

- Batch job card decks
- Data card decks

To ensure that card decks are processed efficiently, you must understand their characteristics and the use of the card reader. The following sections describe which cards you should check before processing a deck through a card reader and how to determine which cards are damaged. Section 6.7.3.2 outlines a procedure for processing a card deck through the card reader.

6.7.1 Distinguishing the Type of Card Deck

Before loading a card deck into the card reader, you should determine:

- Whether the deck is a batch job or a data deck, because their processing requirements differ
- Whether the card reader is set to the correct translation mode

The following sections describe how to make these determinations.

6.7.1.1 Batch Job Card Deck

A batch job card deck consists of three segments:

- Initial cards
- Program cards
- Last card

The initial two cards in a batch job card deck are the $JOB and the $PASSWORD cards. These cards log in the user and the batch job to the system. Following the initial two cards are program cards. Program cards contain instructions that direct the system to libraries, routines, and data needed to complete the batch job. The last card must be either an end-of-job command ($EOJ) card or an end-of-file (EOF) card. Either of these cards tells the system that this is the end of the job.

Checking Input

The system cannot execute the job without $JOB and $PASSWORD cards. If you are given a card deck with these cards omitted, you should return the deck so that the user can insert them.

Since the card deck contains the user's password, you must ensure that it is always handled with care to preserve the security of the user's account.

The last card in the deck must be either an $EOJ or an EOF card.

If the last card is not one of these end cards, you can type an end card on the card punch (12-11-0-1-6-7-8-9 overpunch in column 1) and place it at the end of the deck.

Checking Output

The log file produced by a card reader batch job is queued for printing to the default system printer queue, SYS$PRINT. To have the log file queued to a different queue, the user can specify the /PRINTER qualifier on the $JOB card.

If an error occurs while the system is attempting to validate the $JOB and $PASSWORD cards, OPCOM sends to the card operator an error message that reports the job card and the error.

6.7.1.2 Data Card Deck

A data deck contains data that will be either read by a program or copied to a file for later use. The process that will read the data deck is usually associated with an interactive user at a terminal, or else with a batch job that is submitted by an interactive user. Since the user and process already are logged in to the system, the first card can contain any data the user wants to specify. Then, either the program must read the exact number of cards supplied, or the last card should be an EOF card to inform the program that this is the end of the data deck.

When a user wants a data deck to be read, you should ensure that the user has allocated the card reader. If the card reader is not allocated, the system tries to submit the deck as a batch job and subsequently just flushes the deck through the reader, rejecting the job.

If the program does not read the exact number of cards (as with the COPY command), the EOF card must be the last card in the deck, to inform the program that this is the end of the deck. Without this card, the program waits indefinitely for more cards, and the system prints "card reader offline" messages on the operator's terminal. If the card deck lacks an EOF card, you can type one on a card punch and insert it at the end of the deck.

6.7.2 Setting Card Reader Translation Modes

For the system to read input properly, the card reader must be set to the correct translation mode—the same as the translation mode of the card punch used to prepare the deck. The VMS system supports 026 and 029 card punches.

Therefore, these conditions must exist for you to be able to set the card reader to the correct translation mode:

- The first card in the deck must be the translation mode card.
- You must know the mode in which the cards were punched.

To set the translation mode of the card reader for many decks of the same type, use the SET CARD_READER command. See the *VMS DCL Dictionary* for more information on the SET CARD_READER command. By default, when the system is bootstrapped, the translation mode is set to 029.

6.7.3 Tending the Card Reader

This section describes how to tend the DIGITAL CR-11 card reader. The job of tending the card reader includes:

- Replacing physically defective cards
- Operating the card reader

6.7.3.1 Replacing Physically Defective Cards

Even when the card deck contains all the required cards, the card reader may not be able to read the deck. This problem is usually caused by one or more physically defective cards.

If the deck contains a faulty card, one of the error indicators located on the front panel of the card reader lights up when the card is read. The card reader goes offline, and operator intervention is required to put it back online. Table 6–3 lists the error indicators, the reasons why they may light up, and the operator action required to correct the situation.

6.7.3.2 Operating the Card Reader

The following steps describe how to load and process a card deck through a card reader:

1 Remove the card weight from the input hopper. Place the cards, face down and with column 1 on the left, in the hopper. Ensure that the first card to be read is at the bottom of the hopper.

Do not pack the input hopper so full that the air from the blower cannot riffle the cards. If the cards are packed too tightly, the vacuum picker cannot operate properly.

2 Press the **RESET** button. The **HOPPER CHECK** error indicator and the **STOP** light will go out and the cards will be read.

If the card deck is too large to fit in the input hopper, you can load the excess cards while the reader is operating if you maintain tension on the front portion of the deck.

3 Remove the cards from the output stacker when either the **HOPPER CHECK** error indicator or the **STOP** light is lit.

If you accidentally press the **STOP** button while the card deck is being read, return the last card in the output hopper to the bottom of the input hopper and press the **RESET** button.

4 If the cards are not read properly after the **RESET** button has been pressed, refer to Table 6–3 for recovery procedures.

Table 6–3 Card Reader Errors: Causes and Corrective Actions

Error	Causes	Corrective Action
READ CHECK	Card edges torn Punch in column 0 or 81	Remove the faulty card from the output stacker, duplicate the card, place it in the input hopper, and press the **RESET** button. If READ CHECK occurs for all cards, the read logic of the card reader is malfunctioning.
PICK CHECK	Damage to leading edge Torn webs Cards stapled together	Remove the card from the input hopper, duplicate the faulty card, place the card back in the input hopper, and press the **RESET** button. If there is no evidence of card damage, check for excessive warping of the card deck and/or a buildup of ink glaze on the picker face.
STACK CHECK	Jam in the card track Badly mutilated card	Correct the jam and/or remove the mutilated card from the output stacker, duplicate the card, place it in the input hopper, and press the **RESET** button.
HOPPER CHECK	Input hopper empty Output stacker full	Load the input hopper. Unload the output stacker.

6.7.4 Running the Input Symbiont Interactively

You can run the input symbiont interactively, taking card image input from a VMS RMS file by entering the following commands:

```
$ DEFINE/USER SYS$INPUT filename
$ RUN SYS$SYSTEM:INPSMB
```

Running the input symbiont interactively requires the following:

- CMKRNL privilege
- Read access to the UAF
- Write access to the default directory of the user

All messages are entered to the terminal instead of the card operator.

7 Maintaining System Log Files

The VMS operating system provides several log files that record information about the use of system resources, error conditions, and other system events. These files include the following:

- **System dump file**

  The system dump file assists you in analyzing the cause of the system failure. In the event of a severe system failure, VMS automatically shuts down and produces a crash dump of the state of the system at the time that the error was detected. The DCL command ANALYZE/CRASH_DUMP invokes the System Dump Analyzer (SDA) for analysis of a system dump file (see Section 7.1).

- **Error log file**

  VMS automatically records device and CPU error messages in the error log file. The Error Log Utility invokes the Error Log Report Formatter (ERF), which selectively reports the contents of an error log file (see Section 7.2).

- **Operator log file**

  The Operator Communication Manager (OPCOM) records system events in the operator log file (see Section 7.3).

- **Accounting log file**

  The accounting log file records the use of system resources and is the source of the accounting reports generated by the ACCOUNTING command (see Section 7.4).

If you find errors in the software, submit a Software Performance Report (SPR), along with detailed information about the error condition.

7.1 The System Dump File

The following requirements must be met before the VMS operating system can write a complete dump file:

- Do not halt the system until the console dump messages have been printed in their entirety, and the memory contents have been written to the system dump file. Be sure to allow sufficient time for these events to take place, or make sure that all disk activity has stopped before using the console terminal to halt the system.

- There must be a dump file in the SYS$SYSTEM directory that is named either SYSDUMP.DMP or PAGEFILE.SYS. AUTOGEN automatically creates the SYSDUMP.SYS file if there is enough disk space available.

  If SYS$SYSTEM:SYSDUMP.DMP is present, the system writes dumps to SYSDUMP.DMP. If SYS$SYSTEM:SYSDUMP.DMP is not present, the system writes dumps to SYS$SYSTEM:PAGEFILE.SYS. In this case, PAGEFILE.SYS must be at least 1004 blocks larger than physical memory, and the system parameter SAVEDUMP must be set to 1 (the default is 0). If neither file exists, the system will not generate any dumps.

The size of SYSDUMP.DMP is equal to the physical memory size plus the number of error log buffers plus 1. The number of error log buffers is controlled by the value set for the system parameter ERRORLOGBUFFERS. The range for ERRORLOGBUFFERS is from 2 to 64 with the default set to 4.

- The system parameter DUMPBUG must be set to 1 (the default is 1).

See the *VMS System Dump Analyzer Utility Manual* for more information on the System Dump Analyzer Utility.

7.2 The Error Log File

The system automatically writes error messages to the latest version of a file named SYS$ERRORLOG:ERRLOG.SYS. You can display the information in this file by entering the DCL command ANALYZE/ERROR_LOG.

The Error Logging Facility consists of three parts:

1. A set of executive routines that detect errors and events and write relevant information into error log buffers in memory
2. A process called ERRFMT, which periodically empties the error log buffers, transforms the descriptions of the errors into standard formats, and stores the formatted information in a file on the system disk. (The ERRFMT process is started when the system is booted.)
3. The Error Log Utility, which you invoke by entering the DCL command ANALYZE/ERROR_LOG; it is used to selectively report the contents of an error log file (see the *VMS Error Log Utility Manual*)

The executive routines and the ERRFMT process operate continuously without user intervention. The routines fill the error log buffers in memory with raw data on every detected error and event. When one of the available buffers becomes full, or when a time allotment expires, ERRFMT automatically writes the buffers to ERRLOG.SYS.

Sometimes a burst of errors can cause the buffer to fill up before ERRFMT can empty them. You can detect this condition by noting a skip in the error sequence number of the records reported in the error log reports. As soon as ERRFMT frees the buffer space, the executive routines resume preserving error information in the buffers.

The ERRFMT process displays an error message on the system console and deletes itself if it encounters excessive errors while writing the error log file. To restart the ERRFMT process, first log in to the system manager's account so that you have the required privileges to perform the operation. Then execute the startup command procedure (STARTUP.COM) specifying ERRFMT as the command parameter, as follows:

```
$ @SYS$SYSTEM:STARTUP ERRFMT
```

7.2.1 Using Error Reports

The error reports generated by the Error Log Utility are useful in two basic ways:

1 They aid preventive maintenance by identifying areas within the system that show potential for failure.

2 They aid the diagnosis of a failure by documenting the errors and events that led up to it.

The detailed contents of the reports are most meaningful to DIGITAL field service personnel. However, you can use the reports as an important indicator of the system's reliability. For example, using the DCL command SHOW ERROR, you may see that a particular device is producing a relatively high number of errors. You can then use the Error Log Utility to obtain a more detailed report and decide whether you should consult DIGITAL Field Service. In that case, field service personnel can run diagnostic programs to investigate the device and attempt to isolate the source of the errors.

If a system component does fail, a Field Service representative can study the error reports of the system activity leading up to and including the failure. For example, if a device fails, you can generate error reports immediately after the failure. One report might describe in detail all errors associated with the device that occurred within the last 24 hours; another report might summarize all types of errors for all devices that occurred within the same time period. The summary report can put the device errors into a systemwide context. The Field Service representative can then run the appropriate diagnostic program for a thorough analysis of the failed device. Using the combined error logging and diagnostic information, the Field Service representative can proceed to correct the device.

Error reports allow you to anticipate potential failures. In turn, Field Service personnel rely on the reports as an aid to both preventive and corrective maintenance. Overall, effective use of the Error Log Utility in conjunction with diagnostic programs can significantly reduce the amount of system downtime.

7.2.2 Maintaining the Error Log Files

Because the error log file (SYS$ERRORLOG:ERRLOG.SYS) is a shared file, ERRFMT can write new error log entries while other entries in the same file are being read and reported by the Error Log Utility.

ERRLOG.SYS will increase in size and remain on the system disk until it is explicitly renamed or deleted. Therefore, you must devise a plan for regular maintenance of the error log file.

One method is to rename ERRLOG.SYS on a daily basis. This action causes a new error log file to be created and allows the old file (which was renamed) to be copied to a backup volume where it can be kept as long as needed. For example, you could rename the current copy of ERRLOG.SYS to ERRLOG.OLD every morning at nine o'clock. To free space on the system disk, you could then back up the renamed version of the error log file on a different volume and delete the file from the system disk. Note that you should exercise caution to ensure that error log files are not deleted inadvertently. You may also want to adopt a naming convention for your files that incorporates in the file name a beginning or ending date for the data.

7.2.3 Printing the Error Log Files

The following steps describe how to generate an error log report for all entries in the error log file and how to print the report:

1. Ensure that you have the SYSPRV privilege. You need this privilege to access the error log file.
2. Set your default disk and directory to SYS$ERRORLOG.
3. Examine the error log directory to see which error log file you want to analyze.
4. To obtain a full report of the current error log file, enter the command:

   ```
   $ ANALYZE/ERROR_LOG/OUTPUT=ERRORS.LIS
   ```

5. Print a copy of the report, using the file name specified with the /OUTPUT qualifier:

   ```
   $ PRINT ERRORS.LIS
   ```

Example

```
$ SET PROCESS/PRIV=SYSPRV
$ SET DEFAULT SYS$ERRORLOG
$ DIRECTORY

Directory SYS$SYSROOT:[SYSERR]

ERRLOG.OLD;2  ERRLOG.OLD;1  ERRLOG.SYS;1

Total of 3 files.

$ ANALYZE/ERROR_LOG/OUTPUT=ERRORS.LIS ERRLOG.OLD
$ PRINT ERRORS.LIS
```

The directory command lists all the files in the SYS$ERRORLOG directory. In this example the directory contains three files, two old error log files and the current error log file, ERRLOG.SYS. The ANALYZE/ERROR_LOG command requests that a full report be written to a file called ERRORS.LIS, using the most recent ERRLOG.OLD file as input.

7.3 The Operator Log File

The operator log file (SYS$MANAGER:OPERATOR.LOG) records system events and user requests sent to the operator terminal by the operator communication manager (OPCOM), even when all operator terminals have been disabled. By default, OPCOM is started when your system is booted unless you have a standalone workstation. You use the operator log file to anticipate and prevent hardware and software failures, and to monitor user requests for disk and magnetic tape operations. The following message types appear in the operator's log file:

- Initialization of the operator's log file
- Status reports for devices attached to the system
- Operator terminals enabled and disabled

- Volume mounts and dismounts
- User requests and operator replies
- Changes to system parameters through the SYSGEN Utility
- Security alarm messages
- DECnet–VAX status messages

Example 7–1 illustrates some typical messages found in the operator log file. By regularly examining the operator log file, you can often detect potential problems and take corrective action.

Example 7–1 Sample Operator Log File (SYS$MANAGER:OPERATOR.LOG)

```
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:33:54.07  %%%%%%%%%%%
Operator '_ZEUS$VT333:' has been disabled, user JONES
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:34:15.47  %%%%%%%%%%%
Operator '_ZEUS$VT333:' has been enabled, user SMITH
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:34:15.57  %%%%%%%%%%%
operator status for '_ZEUS$VT333:'
PRINTER, TAPES, DISKS, DEVICES
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:38:53.21  %%%%%%%%%%%
request 1, from user PUBLIC
Please mount volume KLATU in device MTA0:
The tape is in cabinet A
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:39:54.37  %%%%%%%%%%%
request 1 was satisfied.
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:40:23.54  %%%%%%%%%%%
message from user SYSTEM
Volume "KLATU       " mounted, on physical device MTA0:
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:40:38.02  %%%%%%%%%%%
request 2, from user PUBLIC
MOUNT new relative volume 2 () on MTA0:
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:41:07.54  %%%%%%%%%%%
message from user SYSTEM
Volume "KLATU       " dismounted, on physical device MTA0:
15-APR-1986 22:42:14.81, request 2 completed by operator OPA0
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:46:47.96  %%%%%%%%%%%
request 4, from user PUBLIC
_TTB5:, This is a sample user request with reply expected.
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:47:38.50  %%%%%%%%%%%
request 4 was canceled
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:48:21.15  %%%%%%%%%%%
message from user PUBLIC
_TTB5:, This is a sample user request without a reply expected.
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:49:07.90  %%%%%%%%%%%
Device DMA0: is offline.
Mount verification in progress.
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:49:20.22  %%%%%%%%%%%
Mount verification completed for device DMA0:
%%%%%%%%%%%  OPCOM, 15-APR-1986 22:49:37.64  %%%%%%%%%%%
Device DMA0: has been write locked.
Mount verification in progress.
%%%%%%%%%%%  OPCOM, 15-APR-1986 23:33:54.07  %%%%%%%%%%%
message from user NETACP
DECnet shutting down
```

7.3.1 Types of OPCOM Messages

This section describes some of the messages you might find in the operator's log file. See the *VMS Networking Manual* for information about DECnet–VAX status messages.

7.3.1.1 Initialization Messages

When you enter the REPLY/LOG command, the current operator's log file is closed and a new version of that file is created and opened. All subsequent OPCOM messages are recorded in this new log file.

When a new log file is created, the first message recorded in it is an initialization message that tells when and by whom the log file was initialized. This message appears in the following format:

```
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, logfile initialized by operator
operator-name logfile is SYS$MANAGER:OPERATOR.LOG
```

7.3.1.2 Device Status Messages

Some I/O drivers send messages to OPCOM concerning changes in the status of the devices they control. For example, when a line printer goes offline, an OPCOM message is written into the operator's log file at periodic intervals until the device is explicitly returned to online status.

The device status message appears in the operator's log file in the following format:

```
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, device device-name is offline
```

The devices for which this message can appear are card readers, line printers, and magnetic tapes.

7.3.1.3 Terminal Enable and Disable Messages

You designate a terminal as an operator's terminal by entering the REPLY /ENABLE command from the desired terminal. OPCOM confirms the request by displaying the following message at the operator's terminal and in the operator's log file:

```
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, operator enabled, operator terminal-name
```

This message tells you which terminal has been established as an operator's terminal and when it was established.

If a terminal has been designated as an operator's terminal for a particular function, OPCOM displays the name of that function or operator class. For example, if you enter the command REPLY/ENABLE=TAPES, OPCOM displays the following message:

```
%OPCOM, 14-JUN-1982 10:25:35.74, operator enabled, operator TTE1

%OPCOM, 14-JUN-1982 10:25:38.82, operator status for operator TTE1
TAPES
```

OPCOM confirms that the terminal is established as an operator's terminal and indicates that the terminal can only receive and respond to requests concerning magnetic tape-oriented events, such as the mounting and dismounting of tapes.

A terminal that has been designated as an operator's terminal is automatically returned to nonoperator status when the operator logs out. To return the terminal to normal (nonoperator) status without logging off, enter the REPLY /DISABLE command from the terminal. OPCOM confirms that the terminal is no longer an operator's terminal by displaying a message in the following format both at the operator's terminal and in the operator's log file:

```
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, operator disabled, operator terminal-name
```

This message tells you which terminal has been restored to nonoperator status and when the transition occurred.

If a terminal is designated as an operator's terminal and only partial operator status is disabled, OPCOM displays a status message. This message lists which requests the terminal can still receive and respond to. This message is displayed at the operator's terminal and in the operator's log file in the following format:

```
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, operator status for operator terminal-name
status-report
```

For example, suppose you designate a terminal as an operator's terminal that receives messages concerning magnetic tapes and disks, as well as messages intended for the special site-specific operator class known as OPER10. Later, you relinquish the terminal's ability to receive messages concerning tapes. When you enter the REPLY/DISABLE=TAPES command, OPCOM returns the following message:

```
%Opcom, 14-JUN-1988 09:23:45.32, operator status for operator TTA3
 DISKS, OPER10
```

This message tells you that terminal TTA3 still receives and can respond to messages about disks and messages directed to OPER10.

7.3.1.4 Volume Mount and Dismount Messages

Perhaps the widest range of operator messages occur with volume mounts and dismounts. See Example 7–1 for examples of messages relating to mount verification and operator-assisted mounts.

7.3.1.5 User Request and Operator Reply Messages

To communicate with you, the user enters the REQUEST command, specifying either the /REPLY or /TO qualifier.

If the user enters a REQUEST/REPLY command, the request is recorded in the operator's log file in the following format:

```
%OPCOM,dd-mmm-yyyy hh:mm:ss.cc, request request-id from user user-name
__terminal-name:, "message-text"
```

This message tells you which user sent the message, the time the message was sent, the request identification number assigned to the message, the originating terminal, and the message itself.

If the user enters a REQUEST/TO command, the request is recorded in the operator's log file in the format described above, but without a request identification number, as follows:

```
%OPCOM,dd-mmm-yyyy hh:mm:ss.cc, request from user user-name
__terminal-name:, "message-text"
```

For examples of OPCOM messages that result from requests to mount magnetic tapes through the magtape ACP using the REQUEST/BLANK_TAPE and REQUEST/INITIALIZE_TAPE commands, see Chapter 3.

When you respond to a user's request and specify the /TO qualifier, the response is recorded in the operator's log file in the following format:

```
response message
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, request request-id completed by
operator operator-name
```

This message indicates how the operator responded to the user's request, as well as when the response was entered and which operator responded.

When you respond to a user's request and specify the /ABORT qualifier, the response is recorded in the operator's log file in the following format:

```
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, request request-id was canceled.
```

When you respond to a user's request using the /PENDING qualifier, the response is not recorded in the operator's log file because the request has not yet been completed (that is, the request has not been fulfilled or aborted).

When a user enters a REQUEST/REPLY command and you have disabled all terminals as operator's terminals, OPCOM records all subsequent user's requests in the log file in the format shown above, but returns a message to the user indicating that no operator coverage is available.

All other OPCOM responses to REPLY commands, except responses involving the REPLY/ENABLE, REPLY/DISABLE, and REPLY/LOG commands, are not logged in the operator's log file.

7.3.1.6 Sysgen Utility Messages

Users with CMKRNL privilege can use the Sysgen Utility to change system parameters in the running (active) system. Users with the SYSPRV privilege can use the Sysgen Utility to change system parameters in the current system. OPCOM logs all changes made to current system parameters with messages in the following format:

```
%OPCOM, dd-mmm-yyyy hh:mm:ss.cc, message from user user-name
%SYSGEN-I-WRITExxx, system-mode system parameters modified by process ID n
into file y
```

7.3.1.7 Security Alarm Messages

Security alarm messages are included in the operator log file if you enable a security operator terminal and specific alarm events with the SET AUDIT /ENABLE command. Alarm messages are sent to the security operator terminal when the selected events occur. The following is an example of a security alarm OPCOM message:

```
%OPCOM, 15-OCT-1988 12:27:52.26, security alarm on node HERA/
System UAF record modification
```

You can use the command procedure SYS$MANAGER:SECAUDIT.COM to selectively extract information from the operator's log file. Output from SECAUDIT is displayed on SYS$OUTPUT. If you want to write the records to a file, you include the file specification with the /OUTPUT qualifier. The following command writes the records to the file BREAKINS.DAT in your default directory:

```
$ @SYS$MANAGER:SECAUDIT/OUTPUT=BREAKINS.DAT
```

See the *Guide to VMS System Security* for more information on security audits.

7.3.2 Maintaining the Operator Log File

The operator log file normally resides on the system disk in the [SYSMGR] directory. Because this file is in ASCII format, you can print it. You should print copies regularly and retain these copies for reference. Section 7.3.3 describes how to print copies of the operator log file.

A new version of OPERATOR.LOG is created each time the system is rebooted (unless you have a standalone workstation, in which case the operator log file is not started by default). Note that there is one operator log file per node; it is not a shared file. You can also use the DCL command REPLY/LOG to create a new version of the file at any time. The highest version is always the one in use and is inaccessible.

You should devise a plan for regular maintenance of these files. One way is to rename the second-highest version on a daily basis. The procedure for renaming the operator log file is the same as that described in Section 7.2.2 for renaming the error log file. You may want to purge outdated versions of the operator log file on a regular basis. However, you should not delete versions that have not been backed up.

If OPCOM is inadvertently deleted or suspended, or if you want to start it on a standalone MicroVAX processor, use the following method to start it manually:

First, log in to the system manager's account so that you have the required privileges to perform the operation. Then enter the following command to execute the startup command procedure (STARTUP.COM) specifying OPCOM as the command parameter:

```
$ @SYS$SYSTEM:STARTUP OPCOM
```

7.3.3 Printing the Operator Log File

Perform the following operation from an enabled operator terminal to produce a printed copy of the most recent version of the operator log file:

1. Close the current log file and open a new one by entering the following command:

   ```
   $ REPLY/LOG
   ```

2. Set the default to SYS$MANAGER and enter the following command to list all versions of the file:

   ```
   $ DIRECTORY OPERATOR.LOG
   ```

3. Rename the second highest version to OPERATOR.OLD:

   ```
   $ RENAME OPERATOR.LOG;-1 OPERATOR.OLD
   ```

   The version number, -1, specifies that the second highest version of this file is to be renamed. The highest version number is the current operator log file.

4 Obtain a printed copy of the operator log file by entering the following command from an enabled operator terminal:

```
$ PRINT OPERATOR.OLD
```

In the following example, the REPLY/LOG command closes the current log file and opens a new one; the response from OPCOM verifies that a new log file has been opened. The SET DEFAULT command sets the operator default disk to the system disk, thus enabling you to examine the files contained in the directory [SYSMGR]. You can rename the second highest version of the operator log file to OPERATOR.OLD and then enter the PRINT command to request that this version of the operator log file (OPERATOR.OLD) be printed.

```
$ REPLY/LOG

%%%%%%%%%%% OPCOM, 15-APR-1986 12:29:24.52 %%%%%%%%%%%
logfile initialized by operator _MARS$VTA2:
logfile is SYS$MANAGER:OPERATOR.LOG

$ SET DEFAULT SYS$MANAGER
$ DIRECTORY OPERATOR.LOG

Directory SYS$SYSROOT:[SYSMGR]

OPERATOR.LOG;582           OPERATOR.LOG;581

Total of 2 files.

$ RENAME OPERATOR.LOG;-1 OPERATOR.OLD
$ PRINT OPERATOR.OLD
```

7.4 The Accounting Log File

The Accounting facility collects statistics on the use of system resources in an accounting log file SYS$MANAGER:ACCOUNTNG.DAT. This information is used to monitor system activity and charge for the use of system resources. On most VAX processors (with the exception of a standalone MicroVAX processor), the Accounting facility is enabled by default when the system is started. You can modify the SET ACCOUNTING command in the site-specific startup template (SYS$MANAGER:SYSTARTUP_V5.COM) to change the default setting.

READ privilege is required to gain access to the accounting log file. Only a user who has the ACNT privilege can create subprocesses or detached processes in which accounting is disabled. The DCL command RUN /NOACCOUNTING disables all accounting in a created process.

A user with the OPER privilege can selectively disable various kinds of accounting throughout the system by using the DCL command SET ACCOUNTING/DISABLE. See the *VMS DCL Dictionary* for a full description of the SET ACCOUNTING command.

By default, the accounting log file records each of the following activities for all users:

- Batch job termination (BATCH)
- Detached job termination (DETACHED)
- Image activation (IMAGE)
- Interactive job termination (INTERACTIVE)

- Login failures (LOGIN_FAILURE)
- User messages (MESSAGE)
- Network job termination (NETWORK)
- Print jobs (PRINT)
- Process termination (PROCESS)
- Subprocess termination (SUBPROCESS)

Use the SHOW ACCOUNTING command to display which, if any, of these activities are currently being recorded in the accounting log file.

To enable or disable the logging of one or more activities, specify the corresponding keyword in the preceding list with the /ENABLE or /DISABLE qualifier of the SET ACCOUNTING command. (If you do not specify any keywords, the /DISABLE and /ENABLE qualifiers by default disable and enable all the activities listed above.) For example, to enable the recording of login failures, specify the following:

```
$ SET ACCOUNTING/ENABLE=LOGIN_FAILURE
```

To disable the recording of print jobs, specify the following:

```
$ SET ACCOUNTING/DISABLE=PRINT
```

The following list summarizes the characteristics of the accounting log file:

- File name: ACCOUNTNG.DAT (this file is not an ASCII file; hence, it must be formatted before it is printed)
- Residence and directory: SYS$MANAGER
- File organization: sequential
- Record length: variable
- Record types: eight

The eight types of records correspond to the conditions that cause records to be written to the file. These record types are shown in the list that follows. Note that their corresponding codes, as defined in the macro $ACRDEF in SYS$LIBRARY:STARLET.MLB, are shown in parentheses:

- Records written when processes are deleted (ACR$K_PRCDEL)
- Records written when an image terminates (ACR$K_IMGDEL)
- Records written when the system was initialized (ACR$K_SYSINIT)
- Records written when print jobs are queued (ACR$K_PRINT)
- Records written when login failures occurred (ACR$K_LOGFAIL)
- Records written when users' messages are sent to the accounting log file (ACR$K_USER)
- Records that point to the next accounting file (ACR$K_FILE_FL)
- Records that point to the previous accounting file, if any (ACR$K_FILE_BL)

As records are entered in the accounting log file, all but image termination records are immediately flushed to disk. This precaution guarantees the integrity of the file and the completeness of accounting data even if the system fails.

Usually, the current version of the accounting log file is closed at the end of a billing period, and a new version is created and opened. Because the accounting file is always growing, you may want to begin a new accounting file and purge the old version regularly. To begin a new accounting file, enter the DCL command SET ACCOUNTING/NEW_FILE.

If an attempt to write to the accounting log file results in an error, the file is closed automatically and a new copy is created and opened.

7.4.1 Accounting Records

Accounting records contain cumulative accounts of the resources used either by processes or images set up for users, or by print symbionts that print out files for users. Each accounting record contains three fields—user name, UIC, and account name—that identify the user and establish the connection between the accounting record and a user of the system. These fields correspond to similar fields of the user's account record in the user authorization file (UAF).

As system manager, you can use the Accounting Utility to sort, select, and report the accounting records. The reports can provide valuable system management tools. (See the *VMS Accounting Utility Manual.*) Alternatively, by using the detailed accounting records provided by the system, you or perhaps a system programmer can devise programs for reporting on the use of system resources and for billing for their use.

7.4.2 Accounting Report Formats

The Accounting Utility uses the data from the accounting log file to produce accounting reports. Using ACCOUNTING qualifiers, you can produce a variety of report formats, choose how the reports are organized, and select specific report items. Accounting reports can serve as system management tools to learn more about how the system is used, how it performs, and in some cases, how particular individuals use the system. The reports also provide a means of billing users for system resources.

The basic forms of accounting output are listing and binary output. By default, the output is directed to SYS$OUTPUT. However, you can specify an output file with the /OUTPUT qualifier. You can further specify whether the output should be in binary or ASCII format with the /BINARY qualifier. If you specify /BINARY, a binary accounting file is produced that can later be processed with other accounting commands.

The three basic output formats used for displaying data include: brief, full, and summary listings. If you specify /FULL or /SUMMARY, or assume the default (/BRIEF), an ASCII file is produced. The following example illustrates a summary output for the following Accounting command line:

```
$ ACCOUNTING/SUMMARY=(ACCOUNT,USER)/REPORT=(RECORDS,ELAPSED,PROCESSOR)

From:     5-APR-1987 16:33        To:       23-APR-1987  14:18
Account      Username          Total        Elapsed           Processor
                              Records         Time               Time
-----------------------------------------------------------------------
ADMIN        JFUSCIA             128       5 19:43:47.22    0 10:03:58.09
ADMIN        JGREEN               56       0 23:14:23.01    0 00:14:55.17
DECMAIL      POSTOFFICE            2       0 00:04:01.10    0 00:00:02.89
DECNET       NETMGR                1       0 00:01:31.17    0 00:00:02.81
DECNET       NETNONPRIV         2443       2 09:01:15.10    0 01:09:42.61
FIELD        FIELD                31       0 05:18:16.50    0 00:09:41.59
MANUF        BPURPLE              37       1 02:38:45.03    0 02:23:35.42
MANUF        JBROWN              227       4 04:35:07.25    0 04:30:40.60
```

For more information about the Accounting Utility, see the *VMS Accounting Utility Manual.*

Index

Reader's Comments

Guide to Maintaining a
VMS System
AA–LA34A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	□	□	□	□
Completeness (enough information)	□	□	□	□
Clarity (easy to understand)	□	□	□	□
Organization (structure of subject matter)	□	□	□	□
Figures (useful)	□	□	□	□
Examples (useful)	□	□	□	□
Index (ability to find topic)	□	□	□	□
Page layout (easy to find information)	□	□	□	□

I would like to see more/less ______________________________

____

____

What I like best about this manual is ______________________________

____

____

What I like least about this manual is ______________________________

____

____

I found the following errors in this manual:

Page Description

_____ ______________________________

_____ ______________________________

_____ ______________________________

_____ ______________________________

_____ ______________________________

Additional comments or suggestions to improve this manual:

____

____

____

____

I am using **Version** _____ of the software this manual describes.

Name/Title ______________________________ Dept. __________

Company ______________________________ Date __________

Mailing Address ______________________________

______________________________ Phone __________

-- Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here

Reader's Comments

Guide to Maintaining a VMS System
AA–LA34A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	□	□	□	□
Completeness (enough information)	□	□	□	□
Clarity (easy to understand)	□	□	□	□
Organization (structure of subject matter)	□	□	□	□
Figures (useful)	□	□	□	□
Examples (useful)	□	□	□	□
Index (ability to find topic)	□	□	□	□
Page layout (easy to find information)	□	□	□	□

I would like to see more/less ______________________________

____

____

What I like best about this manual is ______________________________

____

____

What I like least about this manual is ______________________________

____

____

I found the following errors in this manual:

Page Description

______ ______________________________

______ ______________________________

______ ______________________________

______ ______________________________

______ ______________________________

Additional comments or suggestions to improve this manual:

____

____

____

____

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

-- Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here