NonStop™ Systems

GUARDIAN™ Operating System Utilities Reference Manual

Operating System Library

82403

NOTICE

Effective with the B00/E08 software release, Tandem introduced a more formal nomenclature for its software and systems.

The term "NonStop 1+™ system" refers to the combination of NonStop 1+ processors with all software that runs on them.

The term "NonStop™ systems" refers to the combination of NonStop II™ processors, NonStop TXP™ processors, or a mixture of the two, with all software that runs on them.

Some software manuals pertain to the NonStop 1+ system only, others pertain to the NonStop systems only, and still others pertain both to the NonStop 1+ system and to the NonStop systems.

The cover and title page of each manual clearly indicate the system (or systems) to which the contents of the manual pertain.

NonStop™ Systems

GUARDIAN™ Operating System Utilities Reference Manual

Abstract
This manual describes the command syntax for commands and programs of the GUARDIAN operating system, including COMINT, FUP, PUP, PERUSE, SPOOLCOM, DCOM, DSAP, and other utilities. This reference manual is for all users of Tandem NonStop systems.

Product Version
GUARDIAN B00

Operating System Version
GUARDIAN B00 (NonStop Systems)

Part No. 82403 A00

March 1985

Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, CA 95014-2599

DOCUMENT HISTORY

Edition	Part Number	Operating System Version	Date
First Edition	82403 A00	GUARDIAN B00	March 1985

New editions incorporate all updates issued since the previous edition. Update packages, which are issued between editions, contain additional and replacement pages that you should merge into the most recent edition of the manual.

NEW AND CHANGED INFORMATION

With this release, the GUARDIAN Operating System Command Language and Utilities Manual (Part No. 82073 F00) is now broken into four manuals:

1. GUARDIAN Operating System Utilities Reference Manual for the NonStop 1+ system (Part No. 82402 A00)

2. GUARDIAN Operating System Utilities Reference Manual for NonStop systems (Part No. 82403 A00)

3. GUARDIAN Operating System User's Guide for the NonStop 1+ system (Part No. 82395 A00)

4. GUARDIAN Operating System User's Guide for NonStop systems (Part No. 82396 A00)

This GUARDIAN Operating System Utilities Reference Manual contains reference material for all users.

The GUARDIAN Operating System User's Guide contains basic task-oriented material for new users. The two new user's guides contain the material that was printed on blue paper in the final edition of the Command Language and Utilities Manual (Sections 1, 2, 3, 5, 6, 7, 9, and 10).

System error and warning messages, which appeared in appendixes of the earlier manuals, now appear in the System Messages Manual for the NonStop 1+ system (Part No. 82408) and the System Messages Manual for NonStop systems (Part No. 82409).

Reference material on the following software products has been added to this manual:

SUBJECT	SOURCE of MATERIAL
PUP	System Operations Manual
PERUSE	Spooler/PERUSE User's Guide
SPOOLCOM	Spooler System Management Guide
DSAP and DCOM	Information on new utilities
PEEK	System Operations Manual

A new, optional disc process (named DP2) is being introduced with this release. The standard disc process for B00 and previous releases of GUARDIAN is now called DP1.

Section 2, "GUARDIAN Command Interpreter (COMINT)"

1. For DP2 volumes, the system subvolume is now $SYSTEM.SYS<nn>, where <nn> is an octal integer. (This is the subvolume containing the operating system image currently in use.)

2. COMINT now has a HELP command that displays the command syntax for all COMINT commands and programs.

3. Two new commands enhance system timekeeping: ADDDSTTRANSITION, for adding Daylight Savings Time transition, and SYSTIMES, for displaying the current date and time as well as the date and time of the last cold load.

4. Nine COMINT commands that perform privileged functions are now separate programs. The new programs are all resident in the $SYSTEM.SYS<nn> subvolume. The nine COMINT programs are:

ADDUSER	DELUSER	RELOAD
BUSCMD	PASSWORD	RPASSWRD
DEFAULT	RCVDUMP	USERS

   The ADDUSER, DEFAULT, DELUSER, PASSWORD, RELOAD, and USERS programs replace the COMINT commands with the same names.

   The other three programs (BUSCMD, RCVDUMP, and RPASSWRD) can be used in place of the original COMINT commands {X|Y}BUS{DOWN|UP}, RECEIVEDUMP, and REMOTEPASSWORD, respectively. However, you can still use the original COMINT commands.

5. The LIGHTS command now allows greater control over the processor lights display. New options are DISPATCHES, SYSPROCS, PAGING, ALL, and BEAT. The default LIGHTS setting is now ON, ALL, BEAT.

6. RELOAD command syntax has changed so that a single command can be used to reload more than one processor.

7. The STATUS command now has a DETAIL option and a new DETAIL listing format.

8. The LOGON command now has a blind password feature. If you enter a comma after your user name when you log on, you are prompted for your password, but what you type is not displayed on the screen.

Section 3, "The File Utility Program (FUP)"

1. New file attributes have been added for DP2 files. These attributes affect the ALTER, CREATE, INFO, SET, SHOW, and RESET commands. The new attributes are:

   [NO] BUFFERED sets the mode of handling write requests to the file: using buffered or write-through cache. The default is NO BUFFERED.

   [NO] AUDITCOMPRESS sets the mode of producing audit-checkpoint messages: using compressed or entire messages. The default is NO AUDITCOMPRESS.

   [NO] VERIFIEDWRITES sets the mode of file writes: verified or unverified. The default is NO VERIFIEDWRITES.

   [NO] SERIALWRITES sets the mode of writes to the mirror: serial or parallel. The default is NO SERIALWRITES.

   MAXEXTENTS <maximum-extents> sets the maximum number of extents that can be allocated for nonpartitioned DP2 files. The value of <maximum-extents> is an integer that is limited by the amount of free space left in the file label; the absolute maximum value is 978 bytes.

   BUFFERSIZE <unstructured-buffer-size> sets the internal transfer size used when accessing an unstructured DP2 file. DP2 rounds <unstructured-buffer-size> upward to the nearest valid DP2 block size: 512, 1024, 2048, or 4096 bytes. The default <unstructured-buffer-size> is 4096 bytes.

2. DP2 requires that file extent sizes be an integral multiple of the block size (for structured files) or the buffer size (for unstructured files). To meet this requirement, DP2 rounds upward any other extent size you specify. (See the SET command for details.)

3. The number of extents that can be allocated for nonpartitioned DP2 files is equal to the value set for the file attribute MAXEXTENTS.

4. FUP DUP has the following new options: PART specifies a new name for a secondary partition of a partitioned file; ALTFILE specifies a new name for an alternate-key file.

5. FUP DUP has the following new options for duplicating files from a DP1 volume to a DP2 volume or vice versa: EXT sets the extent size for the file being duplicated; SLACK, ISLACK, and DSLACK specify slack values to be used when loading data from a source file into a key-sequenced file.

6. FUP INFO now has ROLLFORWARD NEEDED and BROKEN file flags.

7. FUP ALTER now has a RESETBROKEN parameter to reset the file-label BROKEN flag.

8. If you use FUP SET to specify the extent size in units of RECS (records), FUP SHOW displays the extent size in units of RECS.

9. Because data and index blocks must be the same size for DP2 key-sequenced files, DP2 ignores an IBLOCK specification at file-creation time.

Section 4, "The BACKUP and RESTORE Programs"

New utilities are available for backing up and restoring files on the new disc process (DP2). The new BACKUP2 and RESTORE2 programs perform the same general functions as the original BACKUP and RESTORE programs, and their syntax is similar to that of BACKUP and RESTORE. BACKUP2 and RESTORE2, however, have added capabilities for working with DP2 files and for converting files from one disc-process type to another. BACKUP2 and RESTORE2 can copy files between tape and disc as DP1 files or as DP2 files, and both utilities can work with files in the old-backup format as well. See the syntax summaries in Section 4 for details.

Section 5, "Peripheral Utility Program (PUP)"

1. Because you can now configure primary and mirror drives of different subtypes, the syntax of the PUP commands ADDRTOCYL and CYLTOADDR have been changed to specify primary or mirror.

2. The PUP CONSOLE command has a new option, USERMSG [ON | OFF], which lets you turn on and off the logging of user messages to the console log.

3. PUP PRIMARY now has a "!" option that causes the disc process to give away ownership of all four controller paths. Otherwise, PRIMARY gives ownership of only the -P and -M controllers.

4. The PUP COPY command does not allow you to copy a disc to a different disc-process type (DP1 or DP2).

5. The defect log cylinder and the commands that impact it are further explained in the considerations for the PUP FORMAT option. The functions of the PUP FORMAT, NORETAIN option and the PUP SPARE commands have been clarified in this regard.

6. For DP2 files, since the DP2 file directory is dynamically extendible, PUP LABEL now lets you set the cache configuration (that is, how many files you want to fit into a given directory extent) for a volume at the time of volume creation.

7. A new command, PUP LISTCACHE, allows you to examine the current cache configuration for a DP2 volume.

8. A new command, PUP SETCACHE, alters the cache configuration for an in-use DP2 volume.

9. The listing format of PUP LISTDEV has been expanded to display "DPTYPE" to specify DP1- and DP2-configured volumes.

10. The format of the DP2 volume label at this time prevents the use of the PUP REMOVEAUDITED command on a DP2 volume.

11. PUP REFRESH has a new ALL option that writes audit buffers, dirty cache pages, and dirty file control blocks (FCBs) to disc, in addition to main-memory FCBs.

12. A caution has been added to the PUP SPARE command to prevent you from attempting a SPARE operation on a disc that already has a SPARE operation in progress.

13. The UP command has an added consideration that discusses the various causes of UP command failure.

14. The DOWN command now issues a warning and a verification request before executing.

Section 6, "PERUSE"

1. Many new examples have been added to the PERUSE commands.

2. You can now enter multiple PERUSE commands, separated by semicolons, on the same line. The maximum length of the command line is 132 characters.

3. The FIND command now has a BOTH option that allows you to search for a character string in uppercase or lowercase.

4. The display of the JOB command has been expanded to include the closing time of the job, that is, the date and time that the collector finished collecting data from the application.

5. A new command, OPEN, allows you to specify a new spooler supervisor without exiting PERUSE.

Section 7, "SPOOLCOM"

1. A recommendation has been added to the syntax of the SPOOLCOM command to use a local SPOOLCOM to communicate with a remote supervisor.

2. The descriptions of SPOOLCOM commands have been changed to specify their restricted uses by all users and their full uses by super group members.

3. The DEV command has a new HEADER BATCH option which is specially designed for batch processing.

4. The syntax of the LOC command has been broken up into separate sections for each subcommand (STATUS, XREF, BROADCAST, DELETE, and DEV), each with its own considerations and examples.

Section 8, "Disc-Space Analysis and Disc-Compression Utilities (DSAP and DCOM)

Two new DETAIL options have been added:

- BROKEN selects files that are marked as broken due to a detected inconsistency in the file or a read-write I/O error. This option is applicable to DP2 volumes only.
- ROLLFORWARD [NEEDED] selects files audited by the Transaction Monitoring Facility that are marked as needing a rollforward because a crash-recovery operation failed.

Section 9, "Other Utility Programs"

1. PEEK output can now be directed to a file in the format of the EDIT program.
2. These new parameters have been added for PEEK: HELP, TIME, MESSAGES, and INTERRUPTS.
3. You can specify the number of PEEK samples without also specifying the delay interval (the delay interval assumes its default, one second).

Summary of Changes to Spooler Documentation

With the B00 software release, the information formerly in the two spooler manuals, the Spooler/PERUSE User's Guide (Part No. 82093 C00) and the Spooler System Management Guide (Part No. 82094 C00), has been subdivided by audience (general users, system operators and managers, and application programmers) as well as by purpose (task-oriented or reference information).

You can find spooler information in the following new or revised manuals:

Manual	Information
GUARDIAN Operating System User's Guide (Part No. 82395 for NonStop 1+ system Part No. 82396 for NonStop systems)	Introduction to the spooler. How to use SPOOLCOM and PERUSE.
GUARDIAN Operating System Utilities Reference Manual (Part No. 82402 for NonStop 1+ system Part No. 82403 for NonStop systems)	SPOOLCOM and PERUSE syntax, considerations, examples
System Operator's Guide (Part No. 82401)	Information system operators and managers need to start, stop, and control the spooler
System Procedure Calls Reference Manual (Part No. 82359)	Complete syntax and considerations for spooler interface, utility, and print procedures
Spooler Programmer's Guide (Part No. 82394)	Information to help programmers control the spooler from application programs

CONTENTS

FIGURES

TABLES

PREFACE

This first edition of the GUARDIAN Operating System Utilities Reference Manual presents syntax summaries for the principal utilities associated with the GUARDIAN operating system. Because the two Tandem software systems are now documented separately, two versions of this reference manual are available: Part No. 82402 for the NonStop 1+ system and Part No. 82403 for NonStop systems.

This reference manual is intended for all system users, including system operators, system and group managers, and application programmers.

This manual contains much of the reference material from its predecessor, the GUARDIAN Operating System Command Language and Utilities Manual. Material drawn from this source are the sections on COMINT, FUP, BACKUP/RESTORE, and other utilities such as DIVER and DELAY. Reference material on PUP, PEEK, PERUSE, and SPOOLCOM has been added from the System Operations Manual.

Task-oriented material describing the steps involved in logging on to the system and using the GUARDIAN utilities COMINT, FUP, BACKUP/RESTORE, PERUSE, and SPOOLCOM for all users is now in the new GUARDIAN Operating System User's Guide (Part No. 82395 for the NonStop 1+ system and Part No. 82396 for NonStop systems).

Task-oriented material on PUP, PEEK, DSAP, DCOM, and SPOOLCOM for system operators can be found in the System Operator's Guide (Part No. 82400 for the NonStop 1+ system and Part No. 82401 for NonStop systems). This new guide, which replaces the System Operations Manual, contains system information and descriptions of tasks that system operators normally perform.

Before you use this reference manual, you should read the following manuals:

Introduction to Tandem Computer Systems (Part No. 82503)

GUARDIAN Operating System User's Guide (Part No. 82395 for the NonStop 1+ system or Part No. 82396 for NonStop systems)

Programming considerations are discussed in the following manuals:

GUARDIAN Operating System Programmer's Guide (Part No. 82356 for the NonStop 1+ system and Part No. 82357 for NonStop systems)

System Procedure Calls Reference Manual (Part No. 82358 for the NonStop 1+ system and Part No. 82359 for NonStop systems)

Spooler Programmer's Guide (Part No. 82394)

For information on converting files from DP1 format to DP2 format or vice versa, see the DP1/DP2 File Conversion Manual (Part No. 82407).

With this release, the GUARDIAN Operating System Command Language and Utilities Manual (Part No. 82073) becomes obsolete.

SYNTAX CONVENTIONS USED IN THIS MANUAL

The following list summarizes the conventions for syntax notation in this manual.

Notation	Meaning
UPPERCASE LETTERS	Represent keywords and reserved words; you must enter these items exactly as shown.
<lowercase letters>	Angle brackets around lowercase letters represent variables that you must supply.
Brackets []	Enclose optional syntax items. A vertically aligned group of items enclosed in brackets represents a list of selections from which you may choose one or none.
Braces {}	Enclose required syntax items. A vertically aligned group of items enclosed in braces represents a list of selections from which you must choose only one.
Vertical bar |	Separates selections that appear on one line. A horizontally aligned group of items separated by vertical bars and enclosed by brackets or braces represents a list of optional or required syntax items from which you choose one item.
Ellipsis ...	Immediately following a pair of brackets or braces indicates that you can repeat the enclosed syntax items any number of times.
Percent Sign %	Precedes a number in octal notation.
Spaces	May be required or optional: If two syntax items are separated by a space, that space is required between the items. If one of the items is a punctuation symbol, such as a parenthesis or a comma, spaces are optional.
Punctuation (),;!.	Symbols or punctuation not described above must be entered precisely as shown. Any punctuation inside quotation marks must be entered as shown.
RETURN	Indicates pressing the RETURN key.

SECTION 1

INTRODUCTION

This reference manual describes the utility programs associated with the GUARDIAN operating system. Each utility's commands are described separately in the sections of this book. Command entries are presented in alphabetical order. Each command entry contains the complete syntax for that command, considerations for command usage, and annotated examples.

The GUARDIAN operating system utilities described in this book are:

- COMINT, the GUARDIAN command interpreter
- FUP, the GUARDIAN File Utility Program
- BACKUP and RESTORE, BACKUP2 and RESTORE2, utilities for copying disc files onto magnetic tape and for returning them to disc
- PUP, the GUARDIAN Peripheral Utility Program
- PERUSE, a utility for examining and changing attributes of jobs sent to the spooler
- SPOOLCOM, the utility that controls the spooler
- DSAP and DCOM, utilities for disc space analysis and disc compression
- Other utilities, including DELAY and DIVER (used to test NonStop process pairs), ERROR (used to define file-system error numbers), PEEK (used to inform the system manager of the data space usage in a processor), and SPOOL (used to run the spooler supervisor when the spooler is started).

Error messages for all the GUARDIAN utilities are listed and explained in the System Messages Manual.

THE COMMAND INTERPRETER (COMINT)

Section 2 contains the syntax for the commands and programs of the interactive command interpreter, COMINT. Most COMINT commands can be used by any valid user. All users can use COMINT for logging on and off the system, setting up local and remote passwords, starting another COMINT process or other processes, and checking the status of processes. Some COMINT commands, however, are restricted to certain users (such as members of the "super group"--users with group ID 255). A system operator can use COMINT to add or delete users from the system, reload processors, and license privileged programs so that ordinary users can run them.

THE FILE UTILITY PROGRAM (FUP)

Section 3 describes FUP command syntax. You use FUP to create, modify, and manage structured or unstructured files. FUP maintains a set of file-creation attributes that you can set before you create a file or change in your command to create a file. With FUP you can rename, duplicate, and purge files. You can allocate file extents and load data into structured files. You can also control the security attributes of your files, including file ownership, and super-group users can control the licensing of privileged programs.

BACKUP AND RESTORE, BACKUP2 AND RESTORE2

Section 4 describes the syntax for BACKUP and RESTORE and BACKUP2 and RESTORE2, utilities for copying disc files onto magnetic tape and for restoring files from tape to disc. With these utilities, you can select files for backup or restoration based on the date and time of last modification, the file order in the fileset you specify, or the file name. Other capabilities include verifying data written to tape or disc and transferring ownership of restored files. BACKUP and RESTORE work with files in the format of the current disc process (DP1) and in the old backup-format;

BACKUP2 and RESTORE2 work with DP1 files as well as with files in the format of the optional new disc process, DP2. BACKUP and BACKUP2 share a similar set of command parameters, as do RESTORE and RESTORE2. However, both BACKUP2 and RESTORE2 have additional parameters that the original programs do not have. These differences are summarized in Section 4.

THE PERIPHERAL UTILITY PROGRAM (PUP)

Section 5 contains the syntax for PUP commands. A system operator uses PUP to maintain disc drives and other devices. Maintenance functions include initializing disc packs; labeling and bringing up packs; listing bad and spare tracks or sectors on volumes; removing devices from and restoring them to system use.

PERUSE

Section 6 contains the syntax for PERUSE commands. The PERUSE program allows you to examine and change the attributes of jobs sent to the spooler for on-screen viewing or hard-copy printing.

SPOOLCOM

Section 7 contains the syntax for the SPOOLCOM utility. SPOOLCOM provides interactive or noninteractive control of the spooler. SPOOLCOM is primarily for system operators who are members of the super group. Some SPOOLCOM commands, however, can be executed by nonprivileged users.

DISC SPACE ANALYSIS AND DISC COMPRESSION (DSAP AND DCOM)

Section 8 gives the syntax for DSAP and DCOM. These two utilities are designed, respectively, to analyze and to compress disc free space. DSAP provides a range of user options and generates up to seven different reports relevant to the use of disc space. DCOM reduces the number of free-space extents and combines free space into larger extents, thus reducing the effort required to allocate or deallocate space and enabling the allocation of large extent files.

OTHER UTILITIES

Section 9 includes the syntax for these utilities:

- DIVER and DELAY test the function of NonStop process pairs. DELAY delays the execution of the current COMINT process. DIVER downs a processor and prepares it for a reload.
- ERROR displays the explanation for a file-system error number. Enter the number, and ERROR gives a description of the error.
- PEEK gives a snapshot view of the system data-space usage in a processor. This information gives the operator or system manager an idea of the amount of each resource being used.
- SPOOL is used by system operators to run the spooler supervisor when cold or warm starting the spooler.

SECTION 2

GUARDIAN COMMAND INTERPRETER (COMINT)

COMINT (the GUARDIAN command interpreter) is the process that manages interactive communication between you (entering commands at a terminal) and the GUARDIAN operating system.

This section of the manual describes:

- The difference between commands and programs
- Who can use the various COMINT commands and programs (that is, what user ID is required to execute each command or program)
- How to start a COMINT process on another terminal or on another system
- The syntax of all COMINT commands and programs

Syntax descriptions for COMINT commands and programs are given in alphabetical order. Each description defines the syntax elements for that command and includes examples and considerations which affect command operation. This section also gives a summary of COMINT commands and programs grouped according to the class of user who can use them.

COMINT COMMANDS AND PROGRAMS

COMINT is the name of the command interpreter for the GUARDIAN operating system. The file $SYSTEM.SYS<nn>.COMINT contains the COMINT program, where SYS<nn> is the subvolume that contains the operating system image currently in use, and <nn> is an octal integer that identifies this subvolume. COMINT commands are commands executed by COMINT. For example, "the COMINT VOLUME

command" or "the VOLUME command in COMINT" refers to the command named VOLUME that is part of the COMINT program. The COMINT program includes 42 commands.

Nine separate programs are also associated with COMINT. Each of these programs is a a separate program file in the volume $SYSTEM.SYS<nn>. The nine COMINT programs are:

ADDUSER	DELUSER	RELOAD
BUSCMD	PASSWORD	RPASSWRD
DEFAULT	RCVDUMP	USERS

These separate programs:

- Perform privileged operations, such as establishing user IDs and passwords, and reloading processors
- Must be licensed for use by nonprivileged users
- Can run only on the local system
- Each correspond to a different preexisting COMINT command that performs the same function (For example, the ADDUSER command preceded the ADDUSER program; the program accepts both the syntax of the command and the syntax shown in this manual for the program.)

Two terms are used in this manual to refer to COMINT and its related programs. A program is a file containing instructions written in a computer language, such as COBOL or TAL. A process is a program that is running. Although only one program file with a given name exists in any system, there can be many processes running simultaneously from one program file.

You enter COMINT commands and start COMINT processes in the same way--by typing a command at the keyboard when COMINT's colon prompt (:) is displayed, and pressing RETURN. When you enter a COMINT command, the command interpreter interprets and carries out the command. When you enter a command that starts with a program name (such as RELOAD or USERS), COMINT starts the program and then sends a startup message to the new process. The startup message includes the parameters you entered after the program name. The invoked program then carries out your instructions.

Another difference between commands and programs is this: A command to run a program (start a process) can include run options (such as IN <file-name>, OUT <list-file>, and NAME $<process-name>) to direct this run of the process. Most ordinary COMINT commands (such as LOGON and PAUSE) cannot include run options. A complete list of available run options is given in the syntax description for the RUN command.

WHO CAN EXECUTE COMINT COMMANDS?

Most COMINT commands and programs are available to every user. Some commands and programs, however, are restricted so that only certain users can execute them. The summary in this section gives separate lists of restricted commands and programs.

Restricted commands and programs can be used by these two groups:

- Group managers (users that have user ID 255)
- System operators (users that have group ID 255)

(The terms "group ID" and "user ID" are described in the GUARDIAN Operating System User's Guide.)

A super ID user (a user with group ID 255 and user ID 255, and with user name SUPER.SUPER) can execute any commands or programs reserved for group managers or system operators. The super ID user can also:

- Access any file in the system
- Log on as any user without knowing the user's password
- Debug privileged programs
- Run object code stored in any program file in the system
- Suspend or stop any process executing in the system

STARTING A COMINT PROCESS

At system startup, the super ID user cold loads the GUARDIAN operating system from disc. A COMINT process begins execution and controls the terminal named in the cold-load procedure or begins processing an optional command file. This initial COMINT process allows SUPER.SUPER to communicate with the system and to start other COMINT processes at other terminals.

To communicate with the COMINT process at your terminal, you must first log on with the LOGON command. Then you can enter COMINT commands and start processes. By entering a command that begins with the COMINT program name, you can start a COMINT process running on another system or on another terminal in your system. You can also start a COMINT process that uses input from a file rather than from a terminal. (See "COMINT Program.")

COMINT COMMAND SUMMARY

NONRESTRICTED COMMANDS AND PROGRAMS: Any user can execute these commands and programs:

ACTIVATE	Reactivates a previously suspended process.
ALTPRI	Alters a process's execution priority.
ASSIGN	Assigns Tandem file attributes to logical-file descriptions in application programs.
BACKUPCPU	Specifies a backup processor for the current named COMINT process or deletes an existing COMINT backup process.
CLEAR	Deletes attributes previously set by ASSIGN or PARAM commands.
COMINT (program)	Starts another COMINT process.
COMMENT	Introduces a comment line in a COMINT command file.
CREATE	Creates an unstructured disc file.
DEBUG	Puts a process into the debug state.
DEFAULT (program)	Changes the logon default setting for default volume and/or subvolume names; sets default disc file security.
EXIT	When used interactively, stops the current COMINT; when used in a command file, stops execution of commands.
FC	(Fix Command) Modifies or reexecutes the previous COMINT command line.
FILES	Displays the names of files in a subvolume.

→

HELP	Lists the COMINT programs and commands and displays command syntax.
INITTERM	Initializes the home terminal by reinstating the terminal's default SETMODE settings.
LOGOFF	Concludes your session with COMINT.
LOGON	Allows you to begin a session with COMINT.
O[BEY]	Instructs COMINT to execute commands from the file you specify.
PARAM	Assigns a string value to a parameter name. The assigned value can then be passed to an application program.
PASSWORD (program)	Establishes or changes your password.
PAUSE	Causes COMINT to stop prompting for commands or allows another process to control the terminal.
PMSG	Displays the process ID (CPU number and process number) of processes you create or delete.
PPD	Displays the destination-control table.
PURGE	Purges (deletes) a disc file.
REMOTEPASSWORD	Defines a remote password for use in network security.
RENAME	Renames a disc file.
RPASSWORD (program)	Defines a remote password for use in network security.
[RUN[D]]	Runs a program and optionally puts the process started by the command into a debug state.

→

SET	Enables or disables various COMINT environment characteristics. At present, it specifies whether INSPECT is the default debugger for all programs started by COMINT.
SHOW	Displays current values for attributes specified with the SET command.
STATUS	Displays the status of processes running in the system.
STOP	Stops a process.
SUSPEND	Prevents a process from executing until it is reactivated by an ACTIVATE command.
SWITCH	Causes the backup COMINT process to become the primary, and vice versa.
SYSTEM	Changes your current default system.
SYSTIMES	Displays the current date and time, as well as the date and time of the last cold load.
TIME	Displays the current system date and time of day.
USERS (program)	Displays the attributes of users and groups.
VOLUME	Changes your current default volume, subvolume, and/or security.
WAKEUP	Sets COMINT's wakeup mode.
WHO	Displays information about your current default settings.

RESTRICTED PROGRAMS: Only group managers (with user ID 255) can execute these programs:

ADDUSER (program)	Adds new users to a group.
DELUSER (program)	Deletes users from a group.

RESTRICTED COMMANDS AND PROGRAMS: Only system operators (with group ID 255) can execute these commands and programs:

ADDDSTTRANSITION	Adds entries to the daylight savings time transition table.
BUSCMD (program)	Tells the operating system that a bus is (or is not) available for use.
LIGHTS	Controls the processor panel lights.
RCVDUMP (program)	Receives a dump from a halted processor over an interprocessor bus.
RECEIVEDUMP	Receives a dump from a halted processor over an interprocessor bus.
RELOAD (program)	Reloads the operating system image into a processor that was previously halted.
SETTIME	Sets the system date and time-of-day clocks.
{X|Y}BUSDOWN	Tells the operating system that a bus is not available for use.
{X|Y}BUSUP	Tells the operating system that a bus is available for use.

COMINT COMMAND AND PROGRAM DESCRIPTIONS

The rest of this section contains descriptions of the syntax for COMINT commands and programs. Each description contains:

- A summary of the function of the command or program
- The syntax of the command or program, including a description of the syntax parameters and variables
- The listing format used for output (if the command or program produces a display or listing output
- Considerations for the use of the command or program
- Examples of usage of the command or program

File-Name Expansion

COMINT performs file-name expansion for any partial file names you specify for variables such as <file-name> or <list-file>. A partial file name is one that omits some portion of a full file name (that is, system, volume, or subvolume). If you specify a partial file name, COMINT expands the file name using the current default names for system, volume, and subvolume where necessary. For example, if you specify MYFILE, COMINT assumes you mean the file named MYFILE in the current default subvolume, volume, and system.

If you want to specify a file that is not in the current default subvolume (or volume), you must include the appropriate subvolume (or volume) name. Likewise, you must specify the system where a file resides if the file is not on the current default system. For example, if your current defaults are \LOCAL.$WORK.MINE, and you want to specify the file TIMING in the subvolume MINE of the volume $TUNEUP in the remote system \AUTOS, you must specify:

```
\AUTOS.$TUNEUP.MINE.TIMING
```

For more details on file-name expansion, see the GUARDIAN Operating System User's Guide.

ACTIVATE Command

You use the ACTIVATE command to restart a process that was suspended with the SUSPEND command.

The syntax of the ACTIVATE command is:

```
ACTIVATE [ [ \<system-name>.] { $<process-name> }
                              { <cpu>,<pin>     } ]
```

ACTIVATE

entered with no <cpu>,<pin> and $<process-name>, reactivates the last process or process pair suspended from the current COMINT.

\<system-name>

is the name of the system where the process resides.

$<process-name>

is the name of the process or process pair to be reactivated.

<cpu>,<pin>

is the process ID (the CPU number and process number) of the process to be reactivated.

Considerations

- The super ID user can activate any suspended process.
- A group manager can activate any suspended process whose process accessor ID matches the user ID of any member of the group.

- Standard users can activate only those suspended processes whose process accessor IDs match their user IDs. Specifically, your user ID must match the process accessor ID of the process you want to activate. (Process accessor IDs are discussed in the GUARDIAN Operating System User's Guide. Also see the EXPAND Reference Manual for restrictions that apply to processes running in remote systems.)

- A process that is reactivated is placed in the queue of processes ready for execution. In this queue, the process with the highest execution priority is executed first. Thus, a newly reactivated process begins immediate execution only if it has the highest execution priority of the processes in the system that are ready to execute. (You can set execution priority with the ALTPRI command, which is described in this section.)

Examples

Suppose you have started a process named $ORC that is executing as a NonStop process pair. To suspend the process, enter:

```
:SUSPEND $ORC
```

If $ORC is the last process suspended from the current COMINT, then you can reactivate $ORC by entering:

```
:ACTIVATE
```

If $ORC is not the last process you suspended, you must specify the process name (or number), as in this example:

```
:ACTIVATE $ORC
```

If a suspended process does not have a name, you can reactivate it by specifying the <cpu>,<pin> of the process (see the STATUS command). For example, you can reactivate the process with process ID 10,66 by entering:

```
:ACTIVATE 10,66
```

ADDDSTTRANSITION Command

If you are a system operator (with group ID 255), you can use the ADDDSTTRANSITION command to add entries to the daylight savings time (DST) transition table.

The syntax of the ADDDSTTRANSITION command is:

```
ADDDSTTRANSITION <start-date-time> , <stop-date-time> ,
                 <offset>
```

<start-date-time>

is the beginning of the period where <offset> is applicable.

<stop-date-time>

is the end of the period where <offset> is applicable.

The format of <start-date-time> and <stop-date-time> is:

```
{ <month> <day> } <year> , <hour>:<min>[:<sec>] { GMT }
{ <day> <month> }                               { LST }
```

where <month> is the first three letters of the name of the month; GMT stands for Greenwich Mean Time; and LST stands for Local Standard Time.

<offset>

is the difference between standard time and daylight savings time. Specify <offset> as either of these:

- \+ <hour>:<min>
- \- <hour>><min>

The <offset> must be between -8:59 and +8:59.

Consideration

The DST table must be loaded in time sequence with no gaps; that is, except for the first entry, the <start-date-time> of each command must be the same as the <stop-date-time> of the previous ADDDSTTRANSITION command. This implies that many commands will have an <offset> value of zero.

Example

To add two periods of daylight savings time, enter:

```
:ADDDSTTRANSITION 01 APR 1984,2:00 LST, 01 SEP 1984, &
:2:00 LST, 1:00

:ADDDSTTRANSITION 01 SEP 1984,2:00 LST, 01 APR 1985, &
:2:00 LST, 0:00

:ADDDSTTRANSITION 01 APR 1985,2:00 LST, 01 SEP 1985,&
:2:00 LST, 1:00
```

ADDUSER Program

A group manager or a super ID user (someone with group ID 255) can add new users to the system with the ADDUSER program. ADDUSER is a program file residing in subvolume $SYSTEM.SYS<nn>.

The syntax of the command to start an ADDUSER process is:

```
ADDUSER <group-name>.<user-name> , <group-id> , <user-id>
```

NOTE

A command to start an ADDUSER process can include any of the RUN command run options, such as IN <file-name> and OUT <list-file>.

<group-name>.<user-name>

are the names of the new user's group and of the new user, respectively. Each name can contain from one to eight letters or digits, and the first character must be a letter. Lowercase letters are converted to uppercase.

<group-id> , <user-id>

are integers that uniquely identify a group and a user in that group, respectively. Each ID must be in the range from 0 to 255, inclusive. The number 255 is reserved as the <group-id> for system operators and the super ID. The number 255 is also reserved as the <user-id> for group managers and the super ID.

Considerations

- Only a group manager or the super ID user can add new users to a system. SUPER.SUPER can add new users to any group. Group managers can add new users only to their respective groups.

- The super ID user can create a new group by adding a new user with a previously unused <group-name> and <group-id>.
- A group does not have to have a group manager.
- The logon defaults for a new user who was just added to the system are volume $SYSTEM, subvolume NOSUBVOL, and disc file security "AAAA".

Examples

Suppose that there is no <group-name> of MANUF and no <group-id> of 8 in the system. In one ADDUSER command, a user with super ID can create a new group (MANUF) and add a new user (STELLA) with <group-id> 8 and <user-id> 1:

```
:ADDUSER MANUF.STELLA, 8,1
MANUF.STELLA (8,1) HAS BEEN ADDED TO THE USERID FILE.
```

The super ID user can also add a manager to the MANUF group by adding a user with <user-id> 255 and <group-id> 8:

```
:ADDUSER MANUF.HONCHO, 8,255
MANUF.HONCHO (8,255) HAS BEEN ADDED TO THE USERID FILE.
```

The new manager can now add other users to the group:

```
:ADDUSER MANUF.MABEL, 8,2
MANUF.MABEL (8,2) HAS BEEN ADDED TO THE USERID FILE.
:ADDUSER MANUF.FRED, 8,44
MANUF.FRED (8,44) HAS BEEN ADDED TO THE USERID FILE.
```

ALTPRI Command

You use the ALTPRI (alter priority) command to change the execution priority of a process or process pair.

The syntax of the ALTPRI command is:

```
ALTPRI [ [ \<system-name>.] { $<process-name> } ], <priority>
                            { <cpu>,<pin>     }
```

ALTPRI

entered with no <cpu>,<pin> or $<process-name>, alters the execution priority of the last process or process pair started by the current COMINT process.

\<system-name>

is the name of the system where the process is executing. If you do not give a system name, ALTPRI assumes the current default system.

$<process-name>

is the name of the process or process pair whose execution priority is changed.

<cpu>,<pin>

is the process ID (CPU number and process number) of the process whose execution priority is changed.

<priority>

is the new execution priority of the process. Specify <priority> as an integer between 1 and 199, inclusive. The operating system executes higher-priority processes first.

Considerations

- A super ID user can change the priority of any process in the system.
- A group manager can alter the priority of any process whose process accessor ID matches any user ID in the group.
- Standard users can change the priority of processes whose process accessor IDs match their user ID. (For a description of process accessor IDs and creator accessor IDs, see the GUARDIAN Operating System User's Guide.)
- Before increasing the priority of a process, carefully consider the effect the change might have on system performance. For example, assigning a high priority to CPU-bound processes, such as those involving lengthy arithmetic computations, can significantly degrade system performance.

Examples

Suppose that a process named $SLOW is currently executing with an execution priority of 110. You can raise the execution priority of $SLOW to 160 by entering:

```
:ALTPRI $SLOW, 160
```

Suppose you are running a program named $SYSTEM.LENGTHY.ANALYSIS. Output from the STATUS *, USER command shows that the current execution priority of the program is 148:

```
                                    SYSTEM  \MANUF
PID     PRI PFR %WT USERID          MYTERM     PROGRAM FILE NAME

04,054  150 P   003 008,044         $FRED      $SYSTEM .SYS04    .COMINT

05,040  150 P R 003 008,044         $FRED      $SYSTEM .SYS04    .COMINT

08,038  148     000 008,044         $FRED      $SYSTEM .LENGTHY  .ANALYSIS
```

If you don't need the results of this program until tomorrow, you might want to lower the program's execution priority so that it does not compete with programs needed to perform today's work. If most programs in the system execute with priorities greater than 100, you can keep the program from competing with them by entering:

```
:ALTPRI 8,38, 80
```

Output from the STATUS *, USER command shows that LENGTHY.ANALYSIS now has a lower priority:

```
                                        SYSTEM   \MANUF
PID     PRI PFR %WT USERID    MYTERM      PROGRAM FILE NAME

04,054 150 P    003 008,044   $FRED     $SYSTEM .SYS04    .COMINT

05,040 150 P R  003 008,044   $FRED     $SYSTEM .SYS04    .COMINT

08,038 080   R  000 008,044   $FRED     $SYSTEM .LENGTHY  .ANALYSIS
```

ASSIGN Command

You can use the ASSIGN command to assign names of actual files to logical file names used in programs (such as those written in COBOL, FORTRAN, and other programming languages). You can also specify the characteristics of such files.

The syntax of the ASSIGN command is:

```
ASSIGN [<logical-unit> [ , <actual-file-name> ]
       [ , <create-open-spec> ] ...            ]
```

ASSIGN

entered with no following parameters, displays the assigned values for all assignments currently in effect.

<logical-unit>

is the name to which a file name or file attributes are assigned. For <logical-unit>, specify either of these:

*.<logical-file>
<program-unit>.<logical-file>

Both <program-unit> and <logical-file> consist of 1 to 31 alphanumeric, dash (-), or circumflex (^) characters.

Exact meanings of <program-unit>, the asterisk (*), and <logical-file> depend on the application; in general:

- <program-unit> is the name given in the source program to which the file-name assignment is to apply.
- The asterisk (*) applies the assignment to all program units in the object program file being run.
- <logical-file> is the name of the file as given in the source program.

→

For details on FORTRAN or COBOL application treatment of ASSIGN and PARAM commands, see the appropriate language manual.

<actual-file-name>

is the name of the actual physical file. A partial file name is not expanded. However, the application process can expand the file name using the default information passed in the startup message.

If you omit both <actual-file-name> and <create-open-spec>, the current assignment value for <logical-unit> is displayed.

If you omit <actual-file-name> but include <create-open-spec>, blanks are passed in the <actual-file-name> field of the assign message.

<create-open-spec>

is a file-creation or file-open specification that sets certain file attributes. For <create-open-spec>, specify one of these:

<extent-spec>
<exclusion-spec>
<access-spec>
CODE <file-code>
REC <record-size>
BLOCK <block-size>

<extent-spec>

is the size of file extents allocated to the file. For <extent-spec>, specify either of these:

EXT [( ] <pri-extent-size> [ )]
EXT ([<pri-extent-size>] , <sec-extent-size>)

⟶

<pri-extent-size>

is the size of the first file extent (primary extent) allocated to the file. Specify <pri-extent-size> as an integer in the range from 1 to 65,535, inclusive. The interpretation of <record-size>, including the units of memory it represents, is left to the application program.

<sec-extent-size>

is the size of extents allocated to the file after the primary extent (secondary extents). Specify <sec-extent-size> as an integer in the range from 1 to 65,535, inclusive. The interpretation of <record-size>, including the units of memory it represents, is left to the application program.

<exclusion-spec>

is the exclusion mode for <logical-unit>. It determines the circumstances in which other processes can access the file. Specify <exclusion-spec> as one of these:

EXCLUSIVE
SHARED
PROTECTED

- EXCLUSIVE means that no other processes can access <actual-file-name> while the program containing <logical-unit> has the file open.
- SHARED means that other processes can both read and write to <actual-file-name> while the program containing <logical-unit> has the file open.
- PROTECTED means that another process can read, but not write to, <actual-file-name> while the program containing <logical-unit> has the file open.

→

See the ENSCRIBE Programming Manual for more information about exclusion modes.

<access-spec>

is the access mode for <logical-unit>. It specifies the type of file operations that can be performed. Specify <access-spec> as one of these:

I-O
INPUT
OUTPUT

- I-O means that processes can both read the file and write to it.
- INPUT means that processes can only write to the file.
- OUTPUT means that processes can only read the file.

See the ENSCRIBE Programming Manual for more information about access modes.

CODE <file-code>

assigns a file code to <logical-unit>. Specify <file-code> as an integer between 0 and 65,535, inclusive. If <file-code> is omitted, the file code is set to 0. See Table 3-1 for a list of reserved file codes.

REC <record-size>

sets the length of records in <logical-unit>. Specify <record-size> as an integer between 1 and 65,535, inclusive. The interpretation of <record-size>, including the units of memory it represents, is left to the application program.

→

BLOCK <block-size>

sets the size of the data blocks used by <logical-unit>. Specify <block-size> as an integer between 1 and 65,535, inclusive. The interpretation of <block-size>, including the units of memory it represents, is left to the application program.

Considerations

- Use the CLEAR command to delete existing ASSIGN commands.
- All parameters in the ASSIGN command are upshifted (converted to uppercase).
- Normally, the maximum number of assignments that can be in effect at one time is 29. To make more than 29 assignments, specify a larger data space for COMINT with the MEM option when you run the COMINT program. The default data space for COMINT is 11 pages. Each additional page of memory you specify allows you to make 19 additional assignments. Thus, to have from 30 to 49 assignments in effect at once, you must include the MEM 12 option when running the COMINT program.
- COMINT only stores the values assigned by this command and sends the values to requesting processes in the form of assign messages. The ASSIGN command does not create files; interpretation of the assigned values must be made by application programs.
- COMINT creates an assign message for each ASSIGN command in effect. A new process must request its assign messages (if any) following receipt of the startup message. The COBOL and FORTRAN compilers provide the code for this function. TAL programs that use ASSIGN commands must provide their own code for handling assign messages.
- The LOGOFF command deletes existing assignments. However, assignments are retained if you enter a second LOGON command without logging off first; in this case, you are implicitly logged off and logged on again, with all assignments retained. (For more information, see the description of the LOGON command.)

Examples

You can assign an actual file name and file-creation attributes to the logical file PRTFILE in a COBOL program by entering:

```
:ASSIGN PRTFILE, MYFILE, EXT 4096, CODE 9999,EXCLUSIVE,OUTPUT
```

You can assign an actual file name and file-creation attributes to the logical file FT002 in a FORTRAN program by entering:

```
:ASSIGN FT002, DATAFILE, INPUT, EXCLUSIVE
```

You can get a list of the attributes of the logical file PRTFILE by entering:

```
:ASSIGN PRTFILE
```

Here is an example of the information displayed:

```
PRTFILE

          PHYSICAL FILE: MYFILE
          PRIMARY EXTENT: 04096
          FILE CODE: 009999
          EXCLUSION: EXCLUSIVE
          ACCESS: OUTPUT
```

You can list the assigned attributes of all logical files by entering:

```
:ASSIGN
```

Here is an example of the information displayed:

```
PRTFILE

          PHYSICAL FILE: MYFILE
          PRIMARY EXTENT: 04096
          FILE CODE: 009999
          EXCLUSION: EXCLUSIVE
          ACCESS: OUTPUT

FT002

          PHYSICAL FILE: DATAFILE
          EXCLUSION: EXCLUSIVE
          ACCESS: INPUT
```

BACKUPCPU Command

You can start or delete a backup process for the current COMINT process with the BACKUPCPU command.

The syntax of the BACKUPCPU command is:

```
BACKUPCPU [ <cpu-number> ]
```

BACKUPCPU

entered with no <cpu-number>, deletes the last backup process started; has no effect if the current COMINT process has no backup.

<cpu-number>

is the number of the processor where the backup COMINT process is to be started. Specify <cpu-number> as an integer in the range from 0 to 15, inclusive.

Considerations

- A COMINT process must have a name in order to have a backup process. (See the NAME option in the description of the COMINT program or RUN command for more information.)
- The BACKUPCPU command must be entered interactively; it cannot be entered through an OBEY file or input file.

Examples

Fred is running a COMINT process named $CMT1 (this is the process name as displayed by the WHO and PPD commands). When Fred enters the STATUS *, TERM command, he sees that $CMT1 does not have a backup process:

```
                                        SYSTEM  \MANUF
PID     PRI PFR %WT USERID   MYTERM       PROGRAM FILE NAME

00,041 150 P R 000 008,014  $MABEL    $SYSTEM .SYS03    .COMINT
```

Fred starts a backup COMINT process in CPU 1 by entering:

```
:BACKUPCPU 1
```

COMINT displays this response after creating a backup process:

```
BACKUP PROCESS CREATED IN CPU 01
```

Entering "STATUS *, TERM" displays the following information:

```
                                        SYSTEM  \MANUF
PID     PRI PFR %WT USERID   MYTERM       PROGRAM FILE NAME

00,041 150 P R 000 008,014  $MABEL    $SYSTEM .SYS03    .COMINT

01,033 150 P   003 008,014  $MABEL    $SYSTEM .SYS03    .COMINT
```

This display shows that the primary COMINT process (running in processor 0, with process number 41) now has a backup COMINT process (running in processor 1, with process number 33). Both the primary and backup processes have a priority of 150. (See the description of the STATUS command in this section for an explanation of the other categories in the STATUS display.)

To delete this newly created backup process, enter:

```
:BACKUPCPU
```

BUSCMD Program

If you are a system operator (with group ID 255), you can use the BUSCMD program to change the status of the X or Y interprocessor bus. The BUSCMD program file resides in the subvolume $SYSTEM.SYS<nn>. You can use this program as a substitute for the {X|Y}BUSDOWN and {X|Y}BUSUP commands. COMINT still recognizes both the {X|Y}BUSDOWN and the {X|Y}BUSUP commands, and converts them into a BUSCMD process. For details on the syntax and usage of the {X|Y}BUSUP and {X|Y}BUSDOWN commands, see the descriptions of these commands in this section.

The syntax of the command to start a BUSCMD process is:

```
BUSCMD  { X | Y } , { DOWN | UP } , <from-cpu> , <to-cpu>
```

NOTE

A command to start a BUSCMD process can include any run option for the RUN command, such as OUT <list-file>.

{X|Y}

specifies the interprocessor bus (X or Y) whose status is to be changed.

{DOWN|UP}

indicates the interprocessor bus is unavailable (DOWN) or can be used (UP).

<from-cpu>

is a CPU number (from 0 to 15, inclusive) that is one endpoint of the segment of the designated bus in which the operational status is to be changed. Specify -1 to indicate all processors.

→

<to-cpu>

is a CPU number (from 0 to 15, inclusive) that is the other endpoint of the bus segment in which the operational status is to be changed. Specify -1 to indicate all processors.

Examples

The super ID user in a four-processor system can bring down the X bus from processor 1 to all other processors by entering:

```
:BUSCMD X, DOWN, 1, -1
THE X BUS FROM CPU 01 TO 00 HAS BEEN DOWNED.
THE X BUS FROM CPU 01 TO 01 HAS BEEN DOWNED.
THE X BUS FROM CPU 01 TO 02 HAS BEEN DOWNED.
THE X BUS FROM CPU 01 TO 03 HAS BEEN DOWNED.
```

A system operator can bring the Y bus up from processor 1 to processor 2 by entering:

```
:BUSCMD Y, UP, 1, 2
THE Y BUS FROM CPU 01 TO 02 HAS BEEN UPPED.
```

CLEAR Command

With the CLEAR command, you can delete logical-file assignments you made with the ASSIGN command, as well as parameters you set with the PARAM command.

The syntax of the CLEAR command is:

```
CLEAR { ALL [ ASSIGN | PARAM ] }
      { ASSIGN <logical-unit>  }
      { PARAM <param-name>     }
```

ALL

entered without ASSIGN or PARAM, deletes all logical-file assignments and parameters.

ALL ASSIGN

deletes all logical-file assignments made with the ASSIGN command.

ALL PARAM

deletes all parameters set with the PARAM command.

ASSIGN <logical-unit>

deletes <logical-unit>. See the description of the ASSIGN command in this section for information about <logical-unit>.

PARAM <param-name>

deletes <param-name>. For details about <param-name>, see the description of the PARAM command in this section.

Examples

The following command deletes all logical-file assignments made with the ASSIGN command and all parameters set with the PARAM command:

```
:CLEAR ALL
```

To delete all logical-file assignments made with the ASSIGN command, enter:

```
:CLEAR ALL ASSIGN
```

You can delete the logical file PRNTFILE and the information associated with it by entering:

```
:CLEAR ASSIGN PRNTFILE
```

You can delete the parameter SWITCH-1 and its value by entering:

```
:CLEAR PARAM SWITCH-1
```

COMINT Program

Enter the COMINT program name to start a COMINT process on your local system or on a remote system (if your system is part of a network).

The syntax of the command to start a COMINT process is:

```
[ \<system-name>.]COMINT
     [ / <run-option> [ , <run-option> ] ... / ]
     [ <backup-cpu-number> ]
```

\<system-name>

is the name of the system on which COMINT is to run. This parameter is valid only for systems that have a system name, such as those that are part of a network. If you omit \<system-name>, the COMINT process runs on the system from which you issued the COMINT command.

For <run-option> you can specify any one of these:

```
IN <file-name>
OUT <list-file>
NAME [ $<process-name> ]
CPU <cpu-number>
PRI <priority>
NOWAIT
MEM <num-pages>
SWAP <volume-name>
```

IN <file-name>

gives the name of a file (usually a terminal) from which COMINT reads commands. COMINT reads 132-byte records from the specified command file until it encounters an EXIT command or the end of file. Only

→

one command is permitted for each record. The specification for <file-name> can be a nondisc device, a disc file, a disc file in EDIT format, or a process. If you specify a terminal for <file-name>, you must specify the same terminal for the OUT <list-file>.

If you omit this option, the IN file of the current COMINT process is used. (If you are using COMINT interactively, as is the usual case, the IN file is implicitly the terminal where you enter the command.)

OUT <list-file>

gives the name of a file to receive output (unless a command parameter directs output elsewhere). The <list-file> can be a nondisc device (such as a terminal), an existing disc file, or a process. If you specify a terminal for the IN file, you must specify the same terminal for OUT <list-file>.

If you omit the OUT option, the OUT file of the current COMINT process is used. (If you are using COMINT interactively, the OUT <list-file> is implicitly the terminal where you enter the COMINT command.)

NOTE

If the IN and OUT files are the same terminal, that terminal becomes the home terminal for the new COMINT process and its descendants (processes started from that COMINT).

NAME [$<process-name>]

is a symbolic process name to be given to the new COMINT process. Specify $<process-name> as an alphanumeric string from one to five characters long; the first character must be alphabetic. Lowercase letters are converted to uppercase. If you include NAME with no $<process-name>, the system assigns a process name. You must include a NAME parameter if the new COMINT process is to have a backup.

→

CPU <cpu-number>

specifies the processor module where the COMINT process is to execute. Specify <cpu-number> as an integer in the range from 0 to 15, inclusive. If you omit this option, the new COMINT process uses the processor module where the present COMINT process is executing. (Note, however, that if a $CMON (command interpreter monitor) process exists, and if the CPU option is omitted, $CMON can specify a different processor than the one where the current COMINT process is executing. See the GUARDIAN Operating System Programmer's Guide for information about the $CMON process.)

PRI <priority>

sets the execution priority of the COMINT process. Specify <priority> as an integer in the range from 1 to 199, inclusive. If you omit this option, the priority of the new COMINT process is set at one less than the priority of the current COMINT. (However, if a $CMON process exists and the PRI option is omitted, $CMON can specify a different priority. See the GUARDIAN Operating System Programmer's Guide for information about the $CMON process.)

MEM <num-pages>

is the maximum number of data pages to be allocated to the COMINT process. A data page is 1024 words (2048 bytes) of data. Specify <num-pages> as an integer in the range from 11 to 64, inclusive. If you omit this option, 11 pages are allocated. In systems that use the ASSIGN command extensively, you might want to allocate additional data pages so that the COMINT process can store more assignment information.

NOWAIT

means that the current COMINT process does not pause (that is, suspend itself) while the new COMINT process runs. Instead, it returns with a command input prompt as soon as the new COMINT process has read the startup message. Do not use the NOWAIT option if you omit IN and OUT; if you include NOWAIT in this instance, both the new and the current COMINT processes will attempt to communicate with the current terminal, and confusion will result.

SWAP <volume-name>

gives the name of the volume used to hold COMINT's virtual data. Using this parameter alleviates disc contention on $SYSTEM.

<backup-cpu-number>

specifies the processor where the backup for the new COMINT process is to run. Specify <backup-cpu-number> as an integer in the range from 0 to 15, inclusive. If you omit this parameter, no backup process is created. You can specify a backup process only if the COMINT process is both a named process (with the NAME option) and run interactively (that is, with one terminal specified as both the IN file and the OUT file).

Examples

The following command illustrates how to start an interactive COMINT process from a preexisting COMINT process ($TERM1 should not be the current terminal):

```
:COMINT / IN $TERM1, OUT $TERM1, CPU 2, PRI 150, NOWAIT,&
:NAME $C106 / 3
```

After you enter this command:

- A new COMINT process is started; it can accept commands from the terminal $TERM1.
- The name of the new COMINT process is $C106.
- The primary COMINT process for $TERM1 is running in processor 2. The backup process is running in processor 3.
- The execution priority of the new process is 150.
- Because the NOWAIT run option was specified, you can immediately execute another command from the current COMINT without waiting for the new COMINT process to terminate.

The next example shows how to start a COMINT process that uses a command file for input:

```
:COMINT / IN COMFILE, OUT LISTING, NOWAIT /
```

After you enter this command:

- A new COMINT process executes the commands contained in the file COMFILE.
- Output from the new COMINT process is sent to the file LISTING.
- The NOWAIT option means that control of the terminal returns to the original COMINT process while the new COMINT process executes. You are free to enter new commands.

COMMENT Command

Use the COMMENT command to introduce single lines of explanatory text in COMINT command files (files made up of COMINT commands).

The syntax of the COMMENT command is:

```
COMMENT [ <comment-text> ]

<comment-text>

   is any text following the COMMENT command on the same
   line.  COMINT ignores this text when executing commands.
```

Consideration

Although the COMMENT command is normally used in disc files that contain COMINT commands, you can also enter the COMMENT command interactively. For information on executing a COMINT command file, see the description of the OBEY command in this section.

Example

COMMENT command lines explain the purpose of the commands in this COMINT command file:

```
COMMENT  Command File for Displaying Processes
COMMENT  Started by MANUF.FRED on Systems
COMMENT  \MANO, \BIGMAN, and \SANDMAN.
COMMENT  Returns to current default system when done.
SYSTEM \MANO
STATUS *, USER MANUF.FRED
SYSTEM \BIGMAN
STATUS *, USER MANUF.FRED
SYSTEM \SANDMAN
STATUS *, USER MANUF.FRED
SYSTEM
```

CREATE Command

You can use the CREATE command to create an unstructured disc file. An unstructured disc file can contain a data-record structure that the GUARDIAN file system does not recognize, or it can be made up of an array of bytes, without data-record structure. Examples of unstructured files are program code files and text files created with the Tandem EDIT program.

The syntax of the CREATE command is:

```
CREATE <file-name> [ , <extent-size> ]

<file-name>

   is the name of the file to be created.

<extent-size>

   is the size of file extents that can be allocated to the
   file.  Specify <extent-size> as an integer between 1 and
   65,535, inclusive, for the number of data pages
   (2048-byte units of disc storage) in each extent.  The
   disc process can allocate up to 16 extents to any file
   you create with the CREATE command.  The default value
   for <extent-size> is 1.
```

Considerations

- The security of a file you create with the CREATE command is the current default security at the time you enter the CREATE command. See the descriptions of the DEFAULT program and the VOLUME command for information about changing default security.
- You can change the security of a selected file with the FUP SECURE command. See the description of FUP SECURE in Section 3 for more information.

- FUP also has a CREATE command. When you create a file using the CREATE command in COMINT, the first extent allocated (the primary extent) is the same size as extents that are subsequently allocated (secondary extents). With the FUP CREATE command, however, you can specify different sizes for primary and secondary extents. FUP CREATE also allows you to partition files over more than one volume so that a file can contain more than 16 extents. See the description of the FUP CREATE command in Section 3 for more information.

Examples

To create an unstructured disc file named ORANGE in your current default subvolume, enter:

```
:CREATE ORANGE
```

Because the default extent size is 1 page (2048 bytes), the largest size ORANGE can attain (when the maximum of 16 extents is allocated to the file) is 16 pages (32,768 bytes).

To create an unstructured file named BANANA in the subvolume $DUCK.STRUT, with an extent size of 10,240 bytes (5 pages), enter:

```
:CREATE $DUCK.STRUT.BANANA, 5
```

The largest size BANANA can attain (when the maximum of 16 extents is allocated to the file) is 80 pages (163,840 bytes).

DEBUG Command

Use the DEBUG command to inititate debugging for a process that is already executing. Entering the DEBUG command is one way to invoke the DEBUG or the INSPECT debugger.

For information on DEBUG, see the DEBUG Manual. For information about the INSPECT symbolic debugger, see the INSPECT Interactive Symbolic Debugger User's Guide.

The syntax of the DEBUG command is:

```
DEBUG [ <process> ]
      [ , TERM [ \<system-name>.]<terminal-name> ]
```

DEBUG

entered with no <process>, puts into a debug state the last process started from the current COMINT.

<process>

is the process to be debugged. For <process>, specify either of these:

```
[ \<system-name>.]$<process-name>
[ \<system-name>.]<cpu>,<pin>
```

\<system-name>

is the name of the system where the process is executing.

$<process-name>

is the name of the process to be debugged. (See the WHO and PPD commands in this section.)

→

<cpu>,<pin>

is the process ID (the CPU number and process number) of the process to be debugged.

TERM [\<system-name>.]<terminal-name>

designates a new home terminal of the process being debugged. To enter DEBUG or INSPECT commands from a terminal other than the current home terminal of the process being debugged, you must designate that terminal as the new home terminal with the TERM option. You must include the \<system-name> if the new home terminal is connected to a system different from that of the default home terminal.

If you omit the TERM option, DEBUG or INSPECT prompts appear on the current home terminal of the process.

Considerations

- You can only enter the DEBUG command interactively, not in a input file or OBEY file.
- A super ID user can debug any process.
- Only a super ID user can debug privileged processes.
- A group manager can debug any process whose process accessor ID matches the user ID of any user in the group. The manager must also have read access to the program file.
- As a standard user, you can debug only those processes with process accessor IDs that match your user ID. You must also have read access to the program file. (For a description of process accessor IDs, see the GUARDIAN Operating System User's Guide.)
- If you do not have the super ID, the process you name in the DEBUG command does not enter the debug state until it executes its next instruction in the user code space. The process cannot enter the debug state when executing system code.

- DEBUG is the default debugger. When you enter the DEBUG command, DEBUG is invoked unless INSPECT was designated as the debugger for the process when the process was created. You can designate INSPECT as the default debugger in any of these ways:

  --Issue a SET INSPECT ON command, and then run the program.

  --Use the INSPECT option with the RUN command.

  --Include a compiler directive when compiling a COBOL, FORTRAN, or TAL program or when using BINDER.

Examples

Suppose you want to debug an executing process named $ERRER. Enter:

```
:DEBUG $ERRER
```

DEBUG or INSPECT then prompts for commands. (The prompt appears for the debugger that was in effect when the process was created.)

To use the terminal $WHITE to debug the process whose process ID is 8,45, enter:

```
:DEBUG 8,45, TERM $WHITE
```

A DEBUG or INSPECT prompt then appears on terminal $WHITE.

DEFAULT Program

Use the DEFAULT program to set your logon default system, volume, and subvolume names, and to set your logon disc file security. The logon defaults are in effect whenever you log on.

During an operating session, you can change the current defaults with the SYSTEM and VOLUME commands. COMINT uses the current default system, volume, and subvolume names to expand partial file names. When you create a file, it gets the default disc file security unless you explicitly specify a different security. See the GUARDIAN Operating System User's Guide for more information about changing the defaults and about disc file security.

DEFAULT is the name of a program file residing in the subvolume $SYSTEM.SYS<nn>.

The syntax of the command to start a DEFAULT process is:

```
DEFAULT { <default-names> [ , " <default-file-security> " ] }
        {  , " <default-file-security> "                   }
```

NOTE

A command to start a DEFAULT process can include any run option for the RUN command, such as IN <file-name> and OUT <list-file>.

<default-names>

are the volume and/or subvolume names that will be your logon defaults; that is, these are the current defaults when you log on or enter the VOLUME command with no parameters. You can also specify a logon default system name. COMINT uses the current default system, volume, and subvolume names to expand partial file names.

The form of <default-names> is:

```
[ \<system-name>.]<volume-or-subvolume-names>
```

→

\<system-name>

is the system name that will be the default when you log on or enter the SYSTEM command with no parameters. If you omit \<system-name>, your logon default system does not change.

<volume-or-subvolume-names>

is the name of your new logon default volume or subvolume or both. These will be the current defaults when you log on or enter the VOLUME command with no parameters. For <volume-or-subvolume-names>, specify any of these three:

$<default-volume-name>
<default-subvolume-name>
$<default-volume-name>.<default-subvolume-name>

$<default-volume-name>

is the name of your new logon default volume. If you omit $<default-volume-name>, the current default volume is your new logon default.

<default-subvolume-name>

is the name of your new logon default subvolume. If you omit <default-subvolume-name>, the current default subvolume becomes your new logon default.

<default-file-security>

is the default disc file security that will be in effect when you log on or enter the VOLUME command with no parameters. (The current default file security is assigned to newly created disc files unless you explicitly assign a different security setting when you create a file.) For <default-file-security>, specify a string of four characters inside quotation marks; the four characters represent the four security attributes described here.

——>

The form of <default-file-security> is:

<R><W><E><P>

- <R> specifies who can read the file.
- <W> specifies who can write to the file.
- <E> specifies who can execute the file.
- <P> specifies who can purge the file.

Each security attribute (<R>, <W>, <E>, and <P>) can be any of the characters listed in Table 2-1.

File Security

When a file is created, the current default security string (<R><W><E><P>) forms the security for that file. A security string can contain any of the characters listed in Table 2-1. In addition, you can use the FUP SECURE command to specify a hyphen (-) for <W>, <E>, and <P> for a specified set of files. The hyphen indicates that only the super ID user in your local system can perform an action. You cannot, however, specify a hyphen as part of the default security string.

Codes U, C, and N control security on a network; their equivalents for local security are O, G, and A. For example, if you are using a network and you log on to a remote system, you cannot access any of your own files on your local (default) system if they are secured with O, G, or A.

The super ID user in your local system can access any file in the system for any purpose. A remote super ID user (on another system) who has matching remote passwords can access files on your system that have security C, N, or U but cannot access files with security A, G, or O.

Table 2-1. File Security Codes for the DEFAULT Program

Code	Meaning	Explanation
O	Owner	Only the owner can perform the action. ("Owner" means any user who is logged on to the system where the file resides, and who has the same user ID as the owner of the file.)
G	Group	Anyone in the owner's group can perform the action. ("Group" includes any user who is logged on to the system where the file resides, and who has the same <group-id> as the owner of the file.)
A	Anyone	Any user logged on to the system where the file resides can perform the action.
U	User	Only the user can perform the action. ("User" means any user who is logged on to any system in the network, and who has the same user ID as the owner of the file.)
C	Community	Anyone in the owner's community can perform the action. ("Community" includes any user who is logged on to any system in the network, and who has the same <group-id> as the owner of the file.)
N	Network	Any user logged on to any system in the network can perform the action.

Considerations

- Any new logon default values you set with the DEFAULT program do not take effect until the next time you log on, or until you enter the VOLUME command with no parameters.
- The current default settings are often different from your logon default settings. When you log on, your logon default settings are in effect for system (if you are on a network), volume, subvolume, and file security. After you log on, you can change the current default volume or system with the VOLUME or SYSTEM command. Neither of these commands, however, affects your logon default settings. Once you log on again, or enter a VOLUME or SYSTEM command with no parameters, your logon default settings are restored. If you have changed these settings with the DEFAULT program, the new values will be in effect. See the descriptions of the VOLUME and SYSTEM commands for more information.
- When you use the DEFAULT program to set a new default system or volume name, that name must be seven characters or less, excluding the backslash (\) or dollar sign ($).

- When a new user is added, his or her logon defaults are:

  --Volume $SYSTEM

  --Subvolume NOSUBVOL

  --Security "AAAA"

Example

Suppose that you want your logon default volume to be $BIG, and your logon default subvolume to be BAD. You also want your logon default disc file security to allow:

- Any user on any system to read and execute a file with that security
- Only you, the owner of a file, to write to and purge a file with the default security. (Also, you must be logged on to the system where the file resides in order to write to such a file or to purge it.)

To do this, enter:

```
:DEFAULT $BIG.BAD, "NONO"
THE DEFAULT <sys-vol-svol> HAS BEEN CHANGED TO $BIG.BAD
THE DEFAULT <file-security> HAS BEEN CHANGED TO "NONO".
```

To put these new logon defaults into effect, you must either log on again or enter:

```
:VOLUME
```

DELUSER Program

A group manager or a user with super ID (group ID 255) can use the DELUSER program to delete users from a system. DELUSER is the name of a program file residing in subvolume $SYSTEM.SYS<nn>.

The syntax of the command to start a DELUSER process is:

```
DELUSER <group-name>.<user-name>
```

NOTE

A command to start a DELUSER process can include any run option for the RUN command, such as IN <file-name> and OUT <list-file>.

<group-name>.<user-name>

are the group and user names of the user who is to be deleted. Each name can contain from one to eight letters or digits. The first character must be a letter. Lowercase letters in <group-name> and <user-name> are converted to uppercase.

Consideration

A super ID user can delete any user in the system. Group managers can delete only those users who are members of their respective groups.

Example

SUPER.SUPER or the group manager of the BIG group can delete the user BIG.BOZO (with user ID 7,12) by entering:

```
:DELUSER BIG.BOZO
BIG.BOZO (7,12) HAS BEEN DELETED FROM THE USERID FILE.
```

EXIT Command

The EXIT command stops the current COMINT process or COMINT process pair.

The syntax of the EXIT command is:

```
EXIT
```

Considerations

- To use the EXIT command to delete a COMINT process on your home system, you must be logged on to that system. You can, however, use EXIT to delete a COMINT process you started on a remote system without logging on to the remote system.
- If COMINT encounters an EXIT command or an end of file (EOF) when executing an OBEY command file, execution of commands in the file halts, and control returns to the COMINT process where the OBEY command was entered.
- If COMINT encounters an EXIT command when executing commands from an input file (such as an IN file named in the command to start a COMINT process), and COMINT is not executing interactively, the COMINT process is deleted. Control returns to the process from which COMINT was started.

CAUTION

If you delete your last COMINT process, you will be unable to communicate with the system. You can start another COMINT process by entering the COMINT program name at another terminal. (See the syntax description of the COMINT program in this section.) However, if you delete the COMINT process that was started when the system was cold loaded from disc, and there are no other COMINT processes executing in the system, the system must be cold loaded again to reactivate a COMINT process.

Example

If you enter an EXIT command interactively, COMINT asks:

```
ARE YOU SURE YOU WANT TO STOP YOUR <comint-name> CI?
```

(<comint-name> is the process name or process ID (CPU number and process number) of your current COMINT process; if your system is part of a network, <comint-name> also includes the system name.)

If you want to stop the current COMINT, enter Y or y followed by a RETURN. If you enter any other character or only a RETURN, COMINT ignores the EXIT command.

FC Command

Use the FC (Fix Command) command to edit and reenter the previous COMINT command line.

The syntax of the FC command is:

```
FC
```

When you enter the FC command, the command line you entered previous to FC is redisplayed. A period prompt (.) appears below the command.

For example, suppose you are renaming the file JUNK to be TESTFILE. You make a typing error in your RENAME command, and COMINT displays an error message indicating that there is no program file with the name RENEAME, as shown in this example. Rather than retype the entire line, type FC to begin fixing your invalid command:

```
:RENEAME JUNK, TESTFILE
PROGRAM FILE ERROR 011
:FC
:RENEAME JUNK, TESTFILE
.
```

You now see your invalid command redisplayed on the screen. The blank line at the period prompt is an "editing template." In this template, you enter subcommands that make corrections or additions to the command displayed above the template. The rules for entering subcommands and for making corrections and additions are described in the following syntax box.

After you enter the subcommands you want, press the RETURN key. Your edited command and a new editing template then appear. If you want to make more corrections to the edited command, you can enter more subcommands in the editing template and then press RETURN. If the edited command is correct, you can reexecute it by pressing RETURN without entering any subcommands.

To correct the spelling error in this example, you could enter:

```
:RENEAME JUNK, TESTFILE
.   AME
:RENAMEE JUNK, TESTFILE
.      d
:RENAME JUNK, TESTFILE
.
```

Editing Template

The syntax of the editing template is:

```
<subcommand> [ // <subcommand> ] ...

<subcommand> is any of these:

   { R | r } <replacement-string>
   { I | i } <insertion-string>
   { D | d }
   <replacement-string>
   <null-editing-template>
```

The FC command recognizes the beginning of a subcommand as any of these:

- The first nonblank character in the editing template
- The first nonblank character following two slash marks ("//" is a separator--a character sequence that separates subcommands on a given line)
- The first nonblank character following D or d.

Any other character is considered part of a previous subcommand.

A subcommand can immediately follow one or more uppercase or lowercase Ds without being preceded by "//".

→

{ R | r } <replacement-string>

replaces characters in the previous command, starting with the character displayed immediately above the R or r. A <replacement-string> preceded by R or r can be any string of characters, including spaces, and can itself begin with R, I, or D (or r, i, or d). Characters in <replacement-string> replace characters in the previous command on a one-for-one basis.

If the separator character sequence (//) follows this subcommand, all characters in <replacement-string> up to "//", including blanks, replace characters in the previous command. Otherwise, replacement ends with the RETURN.

{ I | i } <insertion-string>

inserts characters into the previous command in front of the character displayed above the I or i. If "//" follows this subcommand, all characters in <insertion-string> up to "//", including blanks, are inserted into the previous command. If no "//" appears, all characters up to the RETURN are inserted.

{ D | d }

deletes characters in the previous command. Any command-line character displayed above a D or d that begins a subcommand in the editing template is deleted.

<replacement-string>

is any subcommand that does not begin with R, I, or D (or r, i, or d). Characters in <replacement-string> replace characters immediately above them in the previous command on a one-for-one basis. For example, a D in <replacement-string> replaces rather than deletes the character displayed above it. Replacement is the default action of the editing template.

→

<null-editing-template>

is an editing template that contains only a RETURN. You can enter a null template by pressing RETURN when the editing template displays only the period prompt. If you enter a <null-editing-template>, the most recently displayed version of the previous command is reexecuted.

Considerations

- You can use the FC command before you are logged on. For example, FC works to correct errors in the LOGON command. If your password contains control characters, however, FC might not edit them correctly.
- You must start typing the FC command in one of the first eight columns of the command line. If the FC command starts after column eight, the command to be fixed might be corrupted.
- If you press the BREAK key before completing an editing template, the FC command is aborted, and the previous command is not reexecuted. The original command is left unchanged.
- If you enter only the subcommand separator (//) in the editing template, followed immediately by a RETURN, the FC command is aborted, and the original command is left unchanged.

Example

The following example demonstrates the use of subcommands D, R, and I and the subcommand separator (//):

```
:COMMENT this are a commnt
:FC
:COMMENT this are a commnt
.              DRis//    Ie
:COMMENT this is a comment
.
```

FILES Command

Use the FILES command to display the names of files contained in one or more subvolumes.

The syntax of the FILES command is:

```
FILES [ / OUT <list-file> / ] [ <subvol-set> ]
```

OUT <list-file>

is the name of the file that receives the output from the FILES command. If you omit this parameter, output is sent to the OUT file that is in effect for the current COMINT (normally, your terminal).

<subvol-set>

is a subvolume or set of subvolumes containing the files whose names are to be listed. If you omit <subvol-set>, FILES lists the files in the current default volume and subvolume. For <subvol-set>, specify either of these:

```
[ \<system-name>.][$<volume-name>.]<subvol-name>
[ \<system-name>.][$<volume-name>.]*
```

\<system-name>

is the name of the system where the files reside. If you omit \<system-name>, the current default system name is assumed.

$<volume-name>

is the name of the volume where the files reside. If you omit $<volume-name>, the current default volume name is assumed.

→

<subvol-name>

is the name of the subvolume where the files reside. If you omit <subvol-name>, the command returns the names of files in the current default subvolume.

*

entered in place of <subvol-name>, lists the names of all files in all subvolumes of the designated or default volume.

Examples

You can list the files in your current default subvolume by entering:

```
:FILES
```

To list the files in all subvolumes on your current default volume, enter:

```
:FILES *
```

To list the files in each subvolume in volume $VOL1, enter:

```
:FILES $VOL1.*
```

You can get a list of the files in subvolume SUBZ on volume $DISC1 of system \ROME by entering:

```
:FILES \ROME.$DISC1.SUBZ
```

HELP Command

Use the HELP command to display the syntax of any COMINT command or of any program associated with COMINT.

The syntax of the HELP command is:

```
HELP [ / OUT <list-file> / ] [<command-name>]
                             [ALL           ]
```

HELP

entered with no following parameters, displays the syntax for the HELP command.

OUT <list-file>

specifies a file to receive the HELP command listing. You can specify a spooler device, a disc file, or a terminal. If you omit this option, the output from the HELP command is sent to the OUT file in effect for the current COMINT (usually your home terminal).

<command-name>

is the name of a COMINT command or program whose syntax is to be listed.

ALL

lists the names of all the COMINT commands and programs.

Examples

To display the syntax of the HELP command, enter:

```
:HELP
```

To display a list of all the COMINT commands and associated programs, enter:

```
:HELP ALL
```

If you have write access to the file SECRET in your current default volume and subvolume ($DATA.SUPER), you can send the syntax of the REMOTEPASSWORD command to the file $DATA.SUPER.SECRET by entering:

```
:HELP / OUT SECRET / REMOTEPASSWORD
```

INITTERM Command

Use the INITTERM (initialize terminal) command to reinstate the default attributes of your home terminal. You normally use this command when an application program has left your terminal in an abnormal state. INITTERM calls the SETMODE procedure, function 28. For information about setting device attributes, see the description of SETMODE settings for terminals in the GUARDIAN Operating System Programmer's Guide and the description of the SETMODE procedure in the System Procedure Calls Reference Manual.

The syntax of the INITTERM command is:

```
INITTERM
```

Consideration

- You cannot use the INITTERM command in a command file; you must enter INITTERM interactively.

LIGHTS Command

System operators (with group ID 255) can use the LIGHTS command to control the processor panel lights (also known as the switch register display). The processor panel lights can be used to display the processor usage during an interval of time specified in the LIGHTS command.

The syntax of the LIGHTS command is:

```
LIGHTS [ ON             ] [ , <sys-option> ... ] [ , BEAT ]
       [ OFF            ] [ , ALL              ]
       [ SMOOTH [<n>]   ]
```

ON

enables the display of processor usage in the processor panel lights. ON is the default if no parameters are specified in the LIGHTS command.

OFF

stops the the lights display and turns off all the processor panel lights.

SMOOTH [<n>]

"smooths" or reduces the variance in the processor usage display; causes the lights to show processor usage over the previous <n> seconds, where <n> is a value between 1 and 60, inclusive. The default value of <n> is 10. Although the processor usage is averaged over <n> seconds, the lights display is refreshed every second. LIGHTS SMOOTH also turns on the processor lights if they are off.

You can include a <sys-option> and the ALL and BEAT parameters only when the lights display is enabled (that is, when LIGHTS ON or LIGHTS SMOOTH is in effect).

→

<sys-option> is any one of these:

DISPATCHES
SYSPROCS
PAGING

- DISPATCHES flashes processor light 13 to indicate passage of 100 dispatches. (After 95 dispatches, light 13 is lit for 5 dispatches.)
- SYSPROCS turns on processor light 14 whenever a system process is executing.
- PAGING turns on processor light 15 whenever a page fault is being processed (that is, the memory manager is waiting for a page transfer from a disc process).

ALL

turns on each preceding <sys-option> (DISPATCHES, SYSPROCS, and PAGING).

BEAT

flashes light 0 once every second to indicate that the processor is functioning.

Figure 2-1 shows the processor panel and each of the processor lights.

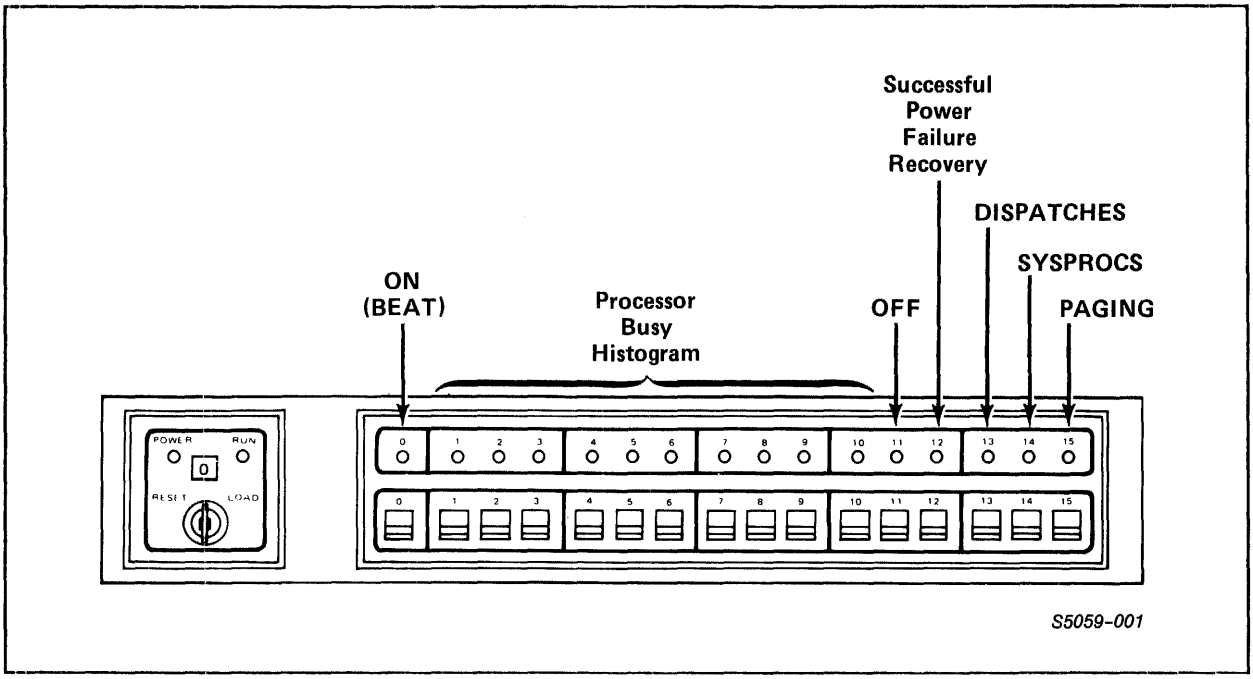


Figure 2-1. Processor Lights

Considerations

- Processor lights are lit for the following reasons:

  --Lights 0 through 10 indicate processor usage. Light 0 is always lit (or flashing if BEAT was specified). One additional light is lit for each 10 percent increase in processor usage.

  --Processor light 11 is always off.

  --Light 12 is lit if the processor successfully recovered from a power failure. When you issue any LIGHTS command, light 12 is turned off in all processors.

  --Lights 13 through 15 are lit for the reasons given in the syntax description for DISPATCHES, SYSPROCS, and PAGING, respectively. Each light is lit for a larger percentage of the time if more processing time is spent on the corresponding activity.

- When the system is cold loaded, the default LIGHTS setting is ON, ALL, BEAT.

Examples

To turn on the processor lights and have them show processor usage smoothed over 10-second intervals, enter:

```
:LIGHTS SMOOTH
```

To change the display to show processor usage for 25-second intervals, enter:

```
:LIGHTS SMOOTH 25
```

To turn on processor light 15 whenever paging activity occurs, enter:

```
:LIGHTS ON, PAGING
```

To turn on light 14 whenever a system process executes, and light 13 after each 100 dispatches, enter:

```
:LIGHTS ,SYSPROCS ,DISPATCHES
```

LOGOFF Command

You terminate a session with COMINT by entering the LOGOFF command.

The syntax of the LOGOFF command is:

```
LOGOFF
```

Considerations

- When you execute the LOGOFF command, you terminate only your session with the current COMINT. Processes that you started can continue to execute after you log off.
- Before logging off, you should stop processes that you are no longer using. Use the STATUS command to get a list of the processes that are executing. Then use the STOP command to stop any unneeded processes.
- Entering the LOGOFF command clears all assignments you made with the ASSIGN, PARAM, and SET commands.
- When you enter the LOGOFF command, the operating system displays the current date and time. If a $CMON process is running, it might display a signoff message (such as "Goodbye.") preceding the date and time display.
- If the ancestor of your current COMINT process is a COMINT process running in another system and you enter the LOGOFF command, the current COMINT process is deleted, and control returns to its ancestor. This message is displayed:

  ```
  EXITING FROM CI ON SYSTEM \<system-name>
  ```

  You must enter another LOGOFF command to terminate communication with the ancestor COMINT.
- If you enter the LOGOFF command while working through a modem on the local system, the modem disconnects.

- After you log off, the current COMINT's process accessor ID and creator accessor ID are set to NULL.NULL (0,0). This is also the setting of the accessor IDs when an interactive COMINT is started, but no user has logged on.

Example

To terminate your session with the current COMINT, enter:

```
:LOGOFF
Goodbye.
15 MAY 1984, 13:31
```

LOGON Command

Use the LOGON command to establish communication with a COMINT process.

To enter COMINT commands interactively, you must first log on. However, when you execute COMINT commands from a command file (with the OBEY command or by specifying a disc file containing COMINT commands as an IN file when you start a new COMINT process), you do not need to include the LOGON command in the file.

The syntax of the LOGON command is:

```
LOGON [ <group-name>.<user-name> [ , [ <password> ] ] ]
```

<group-name>.<user-name>

are the group name and user name of the user who is logging on. The user ID must already be established. To do a blind logon, enter only "LOGON" followed by a RETURN (see "Considerations" and "Examples").

<password>

is the password associated with the user ID. You must include <password> if a password has been established for you. If you have a <password>, you must separate it from your user name with a comma. You can also use the blind password feature by following the comma with a RETURN (see "Considerations" and "Examples"). For more information about passwords, see the description of the PASSWORD program.

Considerations

- When you log on, the operating system displays a logon message that includes:

  --The version number and release date of the operating system

--The number of the processor in which the primary COMINT process is executing

--The number of the processor in which the backup COMINT process is executing (if the process has a backup)

--The current system date and time

- When you enter a full LOGON command, your user name and password are displayed on the terminal screen. If you want to assure security, you can use either the blind logon feature or the blind password feature, both of which are described here. These features work only on terminals that support device SETMODE function 20, which enables or disables system echo mode. You can also use these features on the Operations and Service Processor (OSP) terminal.

  --In a blind logon, <group-name>, <user-name>, and <password> do not display. To do a blind logon, enter "LOGON" and a RETURN, with no user name or password. A question mark (?) prompt appears on the next line. Enter your user name and your password, if you have one. The characters you enter after the ? prompt do not display on the screen.

  --In a blind password, <password> does not display. To use blind password, enter "LOGON" and your group and user name, followed by a comma and RETURN. The prompt "PASSWORD:" appears. Enter your password (or a RETURN if you have no password). Any characters you enter after the PASSWORD: prompt do not display on the screen.

- When you enter the LOGON command, your logon default values are in effect for system, volume, and subvolume names and for disc file security. (See the description of the DEFAULT program for information about logon default values.)

- After you log on, the current COMINT's process accessor ID and creator accessor ID are set to the accessor ID associated with your user name. If your logon fails because you try an illegal or undefined user name or password, the accessor IDs remain set at 0,0. (This is the accessor ID setting when an interactive COMINT is started and no user has logged on.) If three successive attempts to log on at a terminal fail, COMINT ignores attempts to log on from that terminal for one minute.

- The process accessor ID and creator accessor ID of the user logging on are propagated to any descendant processes of COMINT. (For a description of accessor IDs, see the GUARDIAN Operating System User's Guide.)

- If you are logged on with a super ID, you can log on as any user in your system without giving that user's password. Similarly, if you are a group manager, you can log on as any member of your group without giving that user's password.

- Entering the LOGOFF command clears all assignments you made with the ASSIGN and PARAM commands and resets the INSPECT setting to the default value of OFF (see the SET command for information about INSPECT settings). However, if you enter another LOGON command without first logging off, the assignments and parameters are retained, and the current INSPECT setting remains in effect (see the final example).

Examples

User SOFTWARE.JANE, whose password is STAR, can log on interactively by entering:

```
:LOGON SOFTWARE.JANE,STAR
```

Using the blind logon feature, SOFTWARE.JANE can log on in this sequence:

```
:LOGON
?
```

At the question mark (?) prompt, she enters her group and user name and password, which do not appear on the screen:

```
?SOFTWARE.JANE,STAR
```

COMINT displays a logon message, followed by its colon prompt (:). SOFTWARE.JANE can now enter her next command.

Using the blind password feature, SOFTWARE.JANE can log on in this sequence:

```
:LOGON SOFTWARE.JANE,
PASSWORD:
```

At the PASSWORD: prompt, she enters her password, which does not appear on the screen:

```
PASSWORD:STAR
```

A logon message then appears, followed by COMINT's colon prompt (:). SOFTWARE.JANE can now enter commands.

The final example shows how to log on as another user and retain all your current settings. The user MANUF.FRED has logged on and is using the system. Next he logs on with the user name MANUF.MABEL (he must know her password); this LOGON command implicitly logs FRED off. He changes the security of one of Mabel's files, and then he logs on again with his own ID. The final logon also logs off MANUF.MABEL:

```
:LOGON MANUF.MABEL,<password>
:FUP SECURE BIG.FILE, "NNNU"
:LOGON MANUF.FRED,<password>
```

All of Fred's logon defaults remain unchanged, as well as any settings he made with the ASSIGN, PARAM, and SET commands.

OBEY Command

Use the OBEY command to execute a series of COMINT commands contained in a disc file. The file (known as a command file or an OBEY file) can be an unstructured file, such as one created with the EDIT program. Each line in the file contains a COMINT command.

The syntax of the OBEY command is:

```
O[BEY] <command-file>

<command-file>

   is the name of a disc file containing COMINT commands.
```

Considerations

- You cannot use the following commands in an OBEY file: ACTIVATE, ALTPRI, BACKUPCPU, BUSCMD, DEBUG, INITTERM, OBEY, PAUSE, SUSPEND, SWITCH, WAKEUP, XBUSDOWN, XBUSUP, YBUSDOWN, and YBUSUP.
- You can stop the execution of commands in an OBEY file by pressing the BREAK key at the terminal where you entered the OBEY command. COMINT then closes the OBEY file and prompts you for the next command.
- If the processor fails where the primary COMINT process is executing, the backup COMINT process closes the OBEY file and prompts you for a command.

Example

An OBEY file can contain most of the commands that you can enter at COMINT's colon prompt. You can include commands that are to be performed by other programs, such as FUP and PERUSE. Suppose you want to get a daily listing of file activity for all the files in subvolumes $MONEY.MINE, $BILLS.FIRST, and $DATA.SECOND. You could create an EDIT file named ALLFILES in your current default subvolume. ALLFILES should contain the following commands:

```
FUP INFO $MONEY.MINE.*
FUP INFO $BILLS.FIRST.*
FUP INFO $DATA.SECOND.*
```

To execute these commands, enter:

```
:OBEY ALLFILES
```

or

```
:O ALLFILES
```

PARAM Command

You use the PARAM (parameter) command to assign values to parameter names or to obtain a list of the parameter assignments that are currently in effect. Parameters are used to pass information to programs written in COBOL and other languages.

The syntax of the PARAM command is:

```
PARAM [ <param-name> <param-value>
      [ , <param-name> <param-value> ] ... ]
```

PARAM

entered with no <param-name> or <param-value>, displays the names and values of all currently defined parameters.

<param-name>

is the name of the parameter to be assigned a value. <param-name> can contain from 1 to 31 alphanumeric characters, including the circumflex (^) and hyphen (-). Lowercase letters are not converted to uppercase.

<param-value>

is the value assigned to <param-name>. For <param-value> specify either of these:

```
<character-string>
"<character-string>"
```

If you do not enclose <character-string> within quotation marks ("), you cannot include embedded commas (,) or leading or trailing blanks as part of <param-value>.

If you enclose <character-string> within quotation marks ("), all characters, including blanks, between the quotation marks are included as part of <param-value>.

→

You must use double quotation marks ("") to represent a quotation mark within <character-string>. Lowercase letters are not converted to uppercase.

Considerations

- COMINT stores the values of parameters assigned by the PARAM command and sends the values to processes that request parameter values when the processes are started. The interpretation of parameter values is made by the processes that request them.
- To delete existing parameters, use the CLEAR command.
- All parameters are deleted when you enter the LOGOFF command. Parameters and their values are retained, however, if you enter a LOGON command without logging off first.
- COMINT reserves 1024 bytes of internal storage for parameters and their values. This value limits the number and length of parameters in effect. Use the following formula to calculate the number of bytes used by a parameter:

```
 2
+ (number of characters in <param-name>)
+ (number of characters in <param-value>)
```

Examples

You can assign the value ON to the parameter SWITCH-1 and ON to the parameter DEBUG by entering:

```
:PARAM SWITCH-1 ON, DEBUG ON
```

This example assigns a value with leading and trailing blanks and embedded quotes:

```
:PARAM TITLE " ""PIECE OF MY HEART"" "
```

PASSWORD Program

Use the PASSWORD program to select, change, or delete the password you enter each time you log on. (Passwords are optional and might not be required on your system.) PASSWORD is the name of a program file residing in subvolume $SYSTEM.SYS<nn>.

The syntax of the command to run a PASSWORD process is:

```
PASSWORD [ <new-password> ]
```

NOTE

A command to start a PASSWORD process can include any run option for the RUN command, such as OUT <list-file>.

PASSWORD

entered with no following parameters, deletes your current password. No password will be required to log on with your user name.

<new-password>

is the new password that you must enter when you log on. Your <new-password> can include from one to eight characters, except for blanks, commas, and null characters. Lowercase letters are not converted to uppercase.

Considerations

- A new user who was just added to the system has no password.
- Although you can use control characters in your password, entering control characters might cause undesired changes in terminal operation. On some terminals, certain control characters cannot be used in passwords. Refer to the user

documentation for your terminal for information about the effect of control characters on terminal operation.

- If COMINT detects a null character in a password, it rejects the password and returns the message "ILLEGAL PASSWORD." Null characters can be entered inadvertantly by combining certain control characters.

Examples

If you are user 8,44, you can give yourself the password "schubert" (all lowercase) by entering:

```
:PASSWORD schubert
THE PASSWORD FOR USER (008,044) HAS BEEN CHANGED.
```

You must now enter this password whenever you log on.

You can delete your current password without selecting a new one by entering:

```
:PASSWORD
THE PASSWORD FOR USER (008,044) HAS BEEN CHANGED.
```

PAUSE Command

Use the PAUSE command to pause COMINT, that is, to cause COMINT to wait for the completion of another process. When you enter a PAUSE command, the current COMINT stops prompting for commands, thus allowing the other process to gain control of your terminal if it needs to use the terminal for communication. When the other process completes execution or is deleted, the operating system sends a process deletion message to COMINT. Then COMINT regains control of your terminal and displays its prompt (:).

For example, suppose another process controls your terminal, and you press the BREAK key to regain control of the terminal so that you can give commands to COMINT. Then you can use PAUSE to give control back to the other process and stop COMINT from prompting for commands. Control of your terminal returns to COMINT after the process completes or is deleted.

The syntax of the PAUSE command is:

```
PAUSE [ [ \<system-name>.] { $<process-name> }
                           { <cpu>,<pin>     } ]
```

PAUSE

entered with no following parameters, causes COMINT to pause for the last process started from the current COMINT.

\<system-name>

is the name of the system where the process identified by $<process-name> or <cpu>,<pin> is executing. If you omit \<system-name>, the current default system is used.

$<process-name>

is the name of a process that you want COMINT to pause for. COMINT does not prompt for commands until this process sends a process deletion message to COMINT.

→

<cpu>,<pin>

is the CPU number and process number of a process that you want COMINT to pause for. COMINT does not prompt for commands until it receives a process deletion message on this process.

Considerations

- You can only enter the PAUSE command interactively (not through an OBEY file or input file).
- The super ID user can specify any process in a PAUSE command.
- A group manager can pause COMINT for any process whose process accessor ID matches any user ID in the group.
- To list the processes that are currently running on your terminal, use the STATUS *, TERM command. In addition, if a process has a name, you can include its <cpu>,<pin> in a PPD command and obtain its $<process-name>.
- If you enter PAUSE with no parameters when all other processes started from the current COMINT have already been deleted, COMINT waits forever. (No prompt appears on the screen.) To "wake up" COMINT, press the BREAK key.
- Standard users can pause COMINT for only those processes they start (and descendants of those processes). Specifically, you can pause COMINT for another process only if your user ID matches the process accessor ID of the process you want COMINT to wait for. (For a description of accessor IDs, see the GUARDIAN Operating System User's Guide.)

- The current wakeup setting (set with the WAKEUP command) determines the action COMINT takes when it receives a process deletion message.

  --If WAKEUP is set to ON, COMINT regains control of the terminal when any process started from it is deleted.

  --If WAKEUP is set to OFF (the default setting), COMINT regains control of the terminal when the process specified in the PAUSE command is deleted. If no process was specified in the PAUSE command, COMINT regains control of the terminal when the last process started from it is deleted.

- Do not specify a NonStop process pair as $<process-name>. Using the PAUSE command can interfere with NonStop process pairs and can result in errors.

- COMINT attempts to "adopt" the process you name in your PAUSE command by calling the STEPMOM procedure. If this procedure succeeds, COMINT receives the process deletion message when the process completes or is deleted. If this procedure fails, COMINT waits forever (or until you press the BREAK key). For information about the STEPMOM procedure, see the System Procedure Calls Reference Manual.

Example

Suppose you use the BREAK key to temporarily leave a process, such as EDIT, and return control of your terminal to COMINT. Entering PAUSE allows the process, if it is still executing, to regain control of your terminal:

```
*                       The asterisk prompt (*) indicates that
                        EDIT has control of the terminal.

<press BREAK>           Pressing the BREAK key returns control
:                       of the terminal to COMINT.
:PAUSE                  When you enter PAUSE, the EDIT prompt
*                       reappears.
```

PMSG Command

Use the PMSG (process message) command to control logging of creation and deletion messages about processes you create with the RUN command.

The syntax of the PMSG command is:

```
PMSG  { ON | OFF }

ON

   means that whenever a process started with a RUN command
   is created or deleted, a message displays at the current
   COMINT's OUT file (this is usually the home terminal).

OFF (the default setting)

   does not display process creation and deletion messages.
```

The form of process creation messages is:

```
PID: [ \<system-name>.] { $<process-name> }
                        { <cpu>,<pin>     }

\<system-name>

   is the name of the system where the process (represented
   by $<process-name> or by <cpu>,<pin>) is created.

                                                      ——>
```

$<process-name>

is the process name of the newly created process.

<cpu>,<pin>

is the CPU number and process number of the new process.

The form of process deletion messages is:

```
ABENDED: [ \<system-name>.] {$<process-name>}
                            {<cpu>,<pin>    }

STOPPED: [ \<system-name>.] {$<process-name>}
                            {<cpu>,<pin>    }
```

\<system-name>

is the name of the system in which the process abended (abnormally ended) or was stopped.

$<process-name>

is the process name of the process that is to be deleted.

<cpu>,<pin>

is the CPU number and process number of the process that is to be deleted.

Example

The following example shows the type of information you receive when you enter the PMSG ON command:

```
:PMSG ON                Turn on process messages.
:EDIT                   Start an EDIT process.
PID: 08,62              The process ID of the EDIT process
TEXT EDITOR -           is displayed as EDIT signs on.
*
    .
    .
    .
*exit                   When you exit the EDIT process, a
STOPPED: 08,62          process deletion message is displayed.
:
```

PPD Command

Use the PPD (process-pair directory) command to display the names, process IDs, and ancestors of processes currently in the destination-control table (DCT).

The syntax of the PPD command is:

```
PPD [ / OUT <list-file> / ] [ <process> ]
```

PPD

entered with no following parameters, lists all the current entries in the process-pair directory.

OUT <list-file>

is the name of a file to receive the output from the PPD command. If you omit the OUT parameter, the output is sent to the OUT file in effect for the current COMINT (this is usually the terminal where you enter the PPD command).

<process>

is the name of a process whose DCT entry is to be listed. For <process> specify either of these:

```
[ \<system-name>.]$<process-name>
[ \<system-name>.]<cpu>,<pin>
```

\<system-name>

is the name of the system where the process pair (identified by $<process-name> or <cpu>,<pin>) is executing. If you omit \<system-name>, the current default system is assumed.

→

$<process-name>

names the process pair whose DCT entry is to be listed.

<cpu>,<pin>

is the process ID (CPU number and process number) of the primary or backup process whose DCT entry is to be listed.

Listing Format

The PPD command displays information as shown in Figure 2-2.

```
NAME     PID1     PID2     ANCESTOR

<pname>  <pid1>   <pid2>   <ancestor>
   .        .        .         .
   .        .        .         .
   .        .        .         .
```

Figure 2-2. PPD Command Listing Format

The variables that appear in Figure 2-2 are:

<pname> is the name of the process. The name $Z000 identifies the initial COMINT process that was started when the system was cold loaded from disc.

<pid1> is the process ID (CPU number and process number) of the primary process in a process pair.

<pid2>	is the process ID (CPU number and process number) of the backup process in a process pair. If <pid2> is not displayed, the process has no backup.
<ancestor>	is the ancestor of the process, that is, the process that created the process listed in the table. If the ancestor is a named process, this field lists a process name; otherwise, this field contains a process ID. The initial COMINT process that was started when the system was cold loaded from disc has no entry in this field.

If you include \<system-name> in your PPD command, the name of the system is displayed at the beginning of the command listing.

Considerations

- The DCT lists only named processes. To name a process that you start with a RUN command, include the NAME parameter in your command. (For more information about starting and naming processes, see the description of the RUN command in this section.)
- The DCT also lists named processes that do not have backups.

Examples

You can get a listing of all entries in the DCT and send it to the file PROCESS.PAIR by entering:

```
:PPD / OUT PROCESS.PAIR /
```

To view the entry in the DCT for the process $WOW, enter:

```
:PPD $WOW
NAME    PID1    PID2          ANCESTOR

$WOW    04,054  05,009            $Z000
```

PURGE Command

Use the PURGE command to delete disc files.

The syntax of the PURGE command is:

```
PURGE <file-name> [ , <file-name> ]...

<file-name>

   is the name of a permanent or temporary file to be
   deleted.
```

Considerations

- When you use the PURGE command to delete a disc file, the entry for the file is deleted from the file directory in the volume where it resides, and any space previously allocated to that file is made available to other files. Data in the file is not physically deleted from the disc unless you specified the CLEARONPURGE option when you created the file. Files that are physically deleted are overwritten with blank characters. For information about the CLEARONPURGE option, see the description of the CREATE command in Section 3.

- You can purge a file only if it is not currently open, and if you have purge access to the file. See the GUARDIAN Operating System User's Guide for information about file-access restrictions.

- With the FUP PURGE command, you can delete sets of files, in addition to individual files. See the description of the PURGE command in Section 3 for more information.

- If you attempt to use the PURGE command to delete a file that is being audited by the Transaction Monitoring Facility (TMF), the attempt fails, and file-system error 12 (file in use) is returned if there are pending transaction-mode records or file locks. A PURGE attempt of this kind is blocked whether or not the processes that opened the file still exist.

Example

If you have purge access to the files OCT and NOV (in your current default system, volume, and subvolume) and to the file $RECORDS.DUE.PAST, you can purge all of them by entering:

```
:PURGE OCT, NOV, $RECORDS.DUE.PAST
$SWEET.HOME.OCT PURGED
$SWEET.HOME.NOV PURGED
$RECORDS.DUE.PAST PURGED
```

RCVDUMP Program

System operators and super ID users can use the RCVDUMP (receive dump) program to dump the memory from a specific CPU using an interprocessor bus. Your System Operator's Guide describes how to perform bus dumps.

The new RCVDUMP program is a substitute for the RECEIVEDUMP command. COMINT still recognizes the RECEIVEDUMP command and converts it into an RCVDUMP run. RCVDUMP is a program file residing in subvolume $SYSTEM.SYS<nn>.

The syntax for the command to start an RCVDUMP process is:

```
RCVDUMP <dump-file> , <cpu> , { X | Y }
```

NOTE

A command to start a RCVDUMP process can include any run option for the RUN command, such as IN <file-name> and OUT <list-file>.

<dump-file>

is the name of the disc file where the dump is to be written.

<cpu>

is the number of the processor that is to be dumped. Specify <cpu> as an integer in the range from 0 to 15, inclusive. This processor must be primed for a memory bus dump before you attempt the dump.

{ X | Y }

specifies the bus (X or Y) to be used for the dump.

Consideration

If the file <dump-file> does not exist when you run RCVDUMP, a new file named <dump-file> is created. If <dump-file> exists, it must be empty; that is, its end-of-file (EOF) pointer must be set to zero. If <dump-file> is not empty, RCVDUMP aborts. Also, if the empty <dump-file> exists, but its primary and secondary extent sizes are too small to contain the entire dump, the file is purged, and a new <dump-file> with sufficiently large extent sizes is created.

Example

If you are a system operator, you can initiate a dump from processor 6 of your system over interprocessor bus X and send the dump to file $SYSTEM.DUMP.DUMP1 by entering:

```
:RCVDUMP $SYSTEM.DUMP.DUMP1 , 6 , X
CPU 06 HAS BEEN DUMPED TO $SYSTEM.DUMP.DUMP1.
```

RECEIVEDUMP Command

If you are a system operator or the super ID user (that is, if you have group ID 255), you can use the RECEIVEDUMP command to obtain a bus dump of a halted processor. You can also use the equivalent RCVDUMP program, described in this section. Your System Operator's Guide describes how to perform bus dumps.

The syntax of the RECEIVEDUMP command is:

```
RECEIVEDUMP / OUT <dump-file> / <cpu> , <bus>

<dump-file>

   is the name of the disc file to which the dump is sent.

<cpu>

   is the CPU number (from 0 to 15, inclusive) of the
   processor from which the dump is taken.  This processor
   must be primed for a memory bus dump.

<bus>

   is the number of the bus over which the dump is sent:

      0 = the X bus

      1 = the Y bus
```

Consideration

If the file <dump-file> does not exist when you enter the RECEIVEDUMP command, a new file named <dump-file> is created. If <dump-file> exists, it must be empty (that is, its end-of-file (EOF) pointer must be set to zero) when you enter the RECEIVEDUMP command. If <dump-file> is not empty, it is not overwritten; RECEIVEDUMP returns an error message. Also, if the empty

<dump-file> already exists, but its primary and secondary extent sizes are too small to contain the entire dump, the file is purged, and a new <dump-file> with sufficiently large extent sizes is created.

Example

If you are a system operator, you can initiate a dump from processor 4 of your system over interprocessor bus 1 (the Y bus) and send the dump to the file $SYSTEM.DUMP.DUMP2 by entering:

```
:RECEIVEDUMP / OUT $SYSTEM.DUMP.DUMP2 / 4 , 1
CPU 04 HAS BEEN DUMPED TO $SYSTEM.DUMP.DUMP2.
```

RELOAD Program

If you are a system operator or the super ID user (that is, if you have group ID 255), you can use the RELOAD program to reload the remaining processors after cold loading the initial processor. Your System Operator's Guide describes how to cold load systems and reload processors.

RELOAD is the name of a program file residing in subvolume $SYSTEM.SYS<nn>. You can use a remote RELOAD process to load processors in the same remote system. The super ID user can, however, secure the RELOAD program against remote access.

The syntax of the command to start a RELOAD process is:

```
RELOAD [ <cpu-set> [ ; <cpu-set> ...]]
       [ HELP                        ]
```

NOTE

A command to start a RELOAD process can include any run option for the RUN command, such as IN <file-name> or OUT <list-file> (see "Considerations").

RELOAD

entered with no following parameters, displays the help screen for the RELOAD program syntax.

<cpu-set> is defined as:

```
{ <cpu-range>                    }  [ , <option> [ , <option> ]... ]
{ ( <cpu-range> , ... )          }
{ *                              }
```

<cpu-range> can be either of these two:

<cpu-num>
<cpu-num>-<cpu-num>

→

- <cpu-num> is the number of the CPU you want to reload. Specify <cpu-num> as an integer from 0 to 15, inclusive.
- <cpu-num>-<cpu-num> is a range of CPU numbers, where the first <cpu-num> is lower than the second.
- * indicates all CPUs, from 0 to 15, inclusive, that are down, except any <cpu-num> you specify in your RELOAD command.

<option>

is any one of the following three parameters:

$<vol>[.SYS<nn>.OSIMAGE]

names an alternate operating system image to be used for swapping in the reloaded CPU. ($SYSTEM.SYS<nn> is the subvolume that contains the operating system.)

If you include only $<vol>, RELOAD uses the operating system image with the same SYS<nn> subvolume as the CPU doing the reload. If you do not include this parameter at all, the reloaded CPU swaps from the same operating system image as the processor doing the RELOAD (see "Considerations").

<bus>

is the number of the interprocessor bus to be used for reloading the processor:

0 = the X bus

1 = the Y bus

If you omit <bus>, RELOAD uses the bus that is currently selected for transmission by the CPU doing the RELOAD. (Every 11 seconds, transmission switches between buses.)

→

NOSWITCH

overrides the device ownership switches established during configuration. (See "Considerations.")

HELP

displays a help screen summarizing the RELOAD command syntax and giving an example of RELOAD. The help screen is also displayed if you enter RELOAD with no parameters and no IN <file-name> (see "Considerations").

Considerations

- If you try to start a RELOAD process when you are not logged on with group ID 255, this message is displayed:

  INSUFFICIENT CAPABILITY

- If your command repeats a CPU number, RELOAD displays:

  CPU n already specified

  where n is the CPU number you repeated in your RELOAD command.

- The purpose of the alternate OSIMAGE is:

  --To provide additional fault tolerance so that some processors can continue operation if the cold-load volume becomes unavailable

  --To distribute paging activity.

  To name an alternate OSIMAGE file, use the $<vol>[.SYS<nn>.OSIMAGE] parameter. The alternate OSIMAGE file you use should be an exact duplicate of the OSIMAGE file in the CPU doing the reload. RELOAD uses the OSIMAGE in the reloading CPU to initialize the reloaded CPUs. A reloaded CPU accesses the alternate OSIMAGE only when the reloaded CPU faults on an operating system page.

- The alternate OSIMAGE capability is not a substitute for cold loading the system and cannot be used to change the operating system. GUARDIAN assumes that all CPUs have identical operating system images.

- You can use an IN file to enter RELOAD command parameters. For the IN file you can use a disc file but not a terminal. An IN file can contain multiple lines so long as no <cpu-set> is divided between lines. To start a RELOAD process using an IN file, enter RELOAD with only the IN <file-name> specification (see the RUN command in COMINT for details).

- When one of the two processors connected to a controller fails, the other processor gains ownership of the controller. When a failed processor (or one that has not been loaded) is reloaded:

  --If you specify NOSWITCH, no controller ownership switch takes place. The processor that did not fail retains ownership.

  --If you omit NOSWITCH, controller ownership action depends on the type of controller and whether the processor that was reloaded was configured at SYSGEN as the primary processor for the controller. For controllers connected to terminals, serial printers, and data communication lines, no controller ownership switch takes place. For these controllers, problems can arise if an automatic ownership switch occurs during data transmission. For other types of controllers (such as those for discs, tape drives, card readers, and nonserial printers), a controller ownership switch occurs if the reloaded CPU was configured as the primary CPU for the controller.

Examples

To reload processor 1 using the currently selected bus and to prevent switches in device ownership, enter:

```
:RELOAD 1, NOSWITCH
```

The next example uses the Y bus and the alternate operating system image in the file $DATA.SYS02.OSIMAGE to reload processors 2, 4, 5, 6, and 7. The same command then reloads all other downed CPUs using the default operating system image and the currently selected bus:

```
:RELOAD (2, 4-7) , $DATA.SYS02.OSIMAGE , 1; *
```

REMOTEPASSWORD Command and RPASSWRD Program

You can use the REMOTEPASSWORD command or the equivalent RPASSWRD program to establish or delete remote passwords. You must establish remote passwords before you can access remote systems in a network.

To gain access to a remote system, you must have identical remote passwords in effect on both the system where you are logged on and the remote system.

The RPASSWRD process is a substitute for the REMOTEPASSWORD command. COMINT still recognizes the REMOTEPASSWORD command and converts it into an RPASSWRD run. RPASSWRD is the name of a program file that resides in subvolume $SYSTEM.SYS<nn>.

The syntax of the REMOTEPASSWORD command and of the command to start an RPASSWRD process is:

```
{ REMOTEPASSWORD }  [ \<system-name> [ , <password> ] ]
{ RPASSWRD       }
```

NOTE

A command to start an RPASSWRD process can include any run option for the RUN command. The REMOTEPASSWORD command, however, does not accept run options.

\<system-name>

is the name of the system where the remote <password> is to be in effect. If this is your local system, then a remote user with your user ID (normally, you) can access your local system if he or she has set the same <password> for \<system-name> on the remote system.

Conversely, if \<system-name> is the name of a remote system, you can gain access to that system if <password> matches the remote password established for \<system-name> by a person with your user ID on that system.

→

<password>

is the remote password. It can contain from one to eight alphanumeric, nonblank characters. Lowercase letters are not upshifted to uppercase.

CAUTION

If you omit <password>, your remote password for \<system-name> is deleted. If you omit \<system-name>, all of your remote passwords are deleted.

Considerations

- If the RPASSWRD program detects a null character in a remote password, it aborts without changing the remote password. Null characters can inadvertently be included by entering certain control-character combinations at a terminal. See the user documentation for your terminal for a list of null characters that might interfere with your terminal operation.
- See the GUARDIAN Operating System User's Guide for information about the use of remote passwords in network security.

Examples

Suppose that you are user MANUF.FRED on system \ROME, and you need access to files on system \DENHAAG. First, log on to a COMINT running on the \ROME system. Establish the remote passwords YES for the \DENHAAG system and OK for the \ROME system by entering these two commands:

```
:RPASSWRD \DENHAAG, YES
THE \DENHAAG REMOTE PASSWORD FOR MANUF.FRED (8,44) HAS BEEN
  CHANGED.
:RPASSWRD \ROME, OK
THE \ROME REMOTE PASSWORD FOR MANUF.FRED (8,44) HAS BEEN
  CHANGED.
```

Second, a user logged on to the \DENHAAG system whose user ID is identical to yours must establish the identical remote passwords for the \DENHAAG and \ROME systems. This is done by entering the same commands to a COMINT running on the \DENHAAG system:

```
:RPASSWRD \DENHAAG, YES
THE \DENHAAG REMOTE PASSWORD FOR MANUF.FRED (8,44) HAS BEEN
  CHANGED.
:RPASSWRD \ROME, OK
THE \ROME REMOTE PASSWORD FOR MANUF.FRED (8,44) HAS BEEN
  CHANGED.
```

To perform the second step, you normally need to contact the system manager for the \DENHAAG system, who, when logged on with a super ID, can then log on with your user name on the \DENHAAG system and enter a REMOTEPASSWORD or RPASSWRD command.

After these commands are entered, you can gain access to files on \DENHAAG from \ROME, provided that the files are secured so that they would accessible to you if you were logged on at a terminal connected to the \DENHAAG system, and are also secured for network access.

To use the RPASSWRD program to delete the link between \DENHAAG and \ROME for your user ID on the \ROME system, enter:

```
:RPASSWRD \DENHAAG
THE \DENHAAG REMOTE PASSWORD FOR MANUF.FRED (8,44) HAS BEEN
  DELETED.
```

To use the REMOTEPASSWORD command to reestablish the link between \DENHAAG and \ROME, enter:

```
:REMOTEPASSWORD \DENHAAG,YES
THE \DENHAAG REMOTE PASSWORD FOR MANUF.FRED (8,44) HAS BEEN
  CHANGED.
```

RENAME Command

The RENAME command changes the name of an existing disc file.

The syntax of the RENAME command is:

```
RENAME <old-file-name>, <new-file-name>

<old-file-name>

   is the name of the disc file to be renamed.  Partial
   names are expanded for both old and new file names.

<new-file-name>

   is the new name for the file.  For <new-file-name>, you
   can specify the file name, subvolume, or both.
```

Considerations

- You can rename a file only if it is not open with exclusive access, and you either have purge access to the file or are logged on as the super ID user.
- You can use the RENAME command to change a file's subvolume name, but not its volume name. Disc files that are renamed stay on the same disc volume. To change the volume where a file resides, copy the file to a new volume with the File Utility Program (FUP) DUP command, and then delete the original file. For details, see the DUP[LICATE] command in the Section 3.
- If you try to rename a file being audited by the Transaction Monitoring Facility (TMF), the attempt fails and file-system error 80 (operation invalid) is returned. For information about TMF and the AUDIT option, see the Introduction to Transaction Monitoring Facility (TMF).

Example

This command renames the file RECORDS.DATA to STORAGE.OLDDATA:

```
:RENAME RECORDS.DATA, STORAGE.OLDDATA
```

To rename the file MYSTUFF in the current default subvolume to be TRASH in the subvolume EXTRA, enter:

```
:RENAME MYSTUFF, EXTRA.TRASH
```

[RUN[D]] Command

Use the RUN or RUND (run DEBUG) command to run programs. A RUN command must name an object file containing the program you want to run. You can also specify options, such as input and output files to be used by the program; the processor where the program runs; the name and execution priority of the resulting process; and the number of data pages to be used.

You can enter RUN commands either explicitly or implicitly:

- For an explicit RUN command, enter the keyword RUN or RUND followed by the name of the program file.
- For an implicit RUN command, enter the name of the program file, without including the keyword RUN or RUND. The program file must reside in subvolume $SYSTEM.SYSTEM or $SYSTEM.SYS<nn>.

If you enter a command that COMINT does not recognize (one which is not a COMINT command and does not start with RUN or RUND), COMINT assumes you are entering an implicit RUN command and searches for a program file by that name.

RUND puts the process that is started into a debug state before it executes its first instruction. Then either DEBUG or the INSPECT symbolic debugger controls the home terminal of the process. DEBUG is the default debugger. You can specify INSPECT as the debugger with the SET command or with the INSPECT option of the RUN command. See the description of the SET command for more information. For information about DEBUG, see the DEBUG Manual. For information about INSPECT, see the INSPECT Interactive Symbolic Debugger User's Guide.

The syntax of the RUN command is:

```
[ RUN[D] ] <program-file>
           [ / <run-option> [ , <run-option> ]... / ]
           [ <param-string> ]
```

→

<program-file>

is the name of the file containing the object program to be run. Partial file names are expanded using COMINT's current default system, volume, and subvolume names if you enter an explicit RUN command.

If you omit RUN and RUND (thereby giving an implicit RUN command), the operating system first searches for <program-file> in subvolume $SYSTEM.SYSTEM and then in $SYSTEM.SYS<nn>. ($SYSTEM.SYS<nn> is the subvolume where the GUARDIAN operating system image currently in use resides. <nn> is a two-digit octal integer that identifies this subvolume.) If no file by the given name is found, the message "PROGRAM FILE ERROR 11" appears, indicating that the file does not exist.

<run-option> is any one of these:

```
INSPECT { OFF       }
        { ON        }
        { SAVEABEND }
IN [ <file-name> ]
OUT [ <list-file> ]
NAME [ $<process-name> ]
CPU <cpu-number>
PRI <priority>
MEM <num-pages>
NOWAIT
LIB [ <file-name> ]
SWAP [ <file-name> ]
TERM $<terminal-name>
```

```
INSPECT { OFF       }
        { ON        }
        { SAVEABEND }
```

sets the debugging environment for the process being started. ON and SAVEABEND select INSPECT as the debugger, and OFF selects DEBUG (the GUARDIAN debug facility). SAVEABEND is the same as ON except that it automatically creates a save file if the program abends (that is, ends abnormally.)

→

The INSPECT option sets the debugging environment for the process you are starting and for any descendants of that process.

See the DEBUG, SET, and SHOW commands in this section and the INSPECT Interactive Symbolic Debugger User's Guide for more information.

IN [<file-name>]

is the new process's input file. This file name is sent to the new process in its startup message. If you do not include IN <file-name>, the new process uses the IN file in effect for the current COMINT (usually your home terminal). If you include IN with no <file-name>, blanks are sent as the name of the input file.

OUT [<list-file>]

is the new process's output file. If you omit OUT <list-file>, the new process uses the OUT file in effect for the current COMINT (usually your home terminal). If you include OUT with no <list-file>, blanks are sent as the name of the output file.

NAME [$<process-name>]

is the symbolic name you are assigning to the new process. Specify $<process-name> as an alphanumeric string of one to five characters, the first of which must be alphabetic. If you omit this parameter, the new process is not named and has only a process ID (a CPU number and process number). If you include NAME with no $<process-name>, the system generates a name for the new process. The process's name appears in the destination-control table (DCT) part of the process-pair directory (PPD).

→

CPU <cpu-number>

is the number of the processor where the new process is to execute. Specify <cpu-number> as an integer in the range from 0 to 15, inclusive. If you omit this option, the new process executes in the same processor as COMINT. (However, if a $CMON process exists and the CPU option is omitted, $CMON might specify a processor other than the one where the current COMINT process is executing. See the GUARDIAN Operating System Programmer's Guide for information about $CMON processes.)

PRI <priority>

is the execution priority of the new process. If you omit this option, the new process is given a priority of 1 less than COMINT's priority. Specify <priority> as an integer in the range from 1 to 199, inclusive. (Processes with higher numbers are executed first.) If you specify a priority greater than 199, the process runs at priority 199. (However, if a $CMON process exists and you omit the PRI option, $CMON might specify a different priority. See the GUARDIAN Operating System Programmer's Guide for information about $CMON processes.)

MEM <num-pages>

is the maximum number of virtual data pages to be allocated for the new process. Specify <num-pages> as an integer in the range from 1 to 64, inclusive. If you omit this option, or if <num-pages> is less than the compile-time value, the compilation value is used.

NOWAIT

means that COMINT does not wait while the program runs but returns a command input prompt after sending the startup message to the new process. If you omit this option, COMINT pauses while the program runs.

→

LIB [<file-name>]

selects a "user library" file of object routines that are to be searched before the system library file for satisfying external references in the program being run. If you give the name of a library file, the program uses that library until you select another library file, or until you specify LIB without a file name. If you do not give a file name, LIB deletes the previous selection.

To run a program file with a user library, you must have write access to the program file; the library file name is written into the program's object-file header at run time.

To run the program again with the same library, you can omit the LIB parameter. To run the program again with no library (or with a different library), include LIB / (or LIB <file-name>). (Also, see "Considerations.")

SWAP [<file-name>]

specifies the name of the file used to hold the process's virtual data. When a process is executing, the system allocates a temporary file on the same volume as the program file for swapping the data stack. When the process terminates, the temporary swap file is automatically purged. If the swap file has a permanent name, however, it is not purged.

With the SWAP parameter, you can:

- Specify a permanent file name--the file contains an image of the data stack when the process terminates
- Specify a different volume for the swap file (by specifying only a volume name)--this is useful when the program file's volume is full or busy

The SWAP option also specifies the default volume for extended data segments (see the GUARDIAN Operating System Programmer's Guide for more details).

→

The SWAP option should be used for debugging purposes only.

TERM $<terminal-name>

specifies the home terminal for the new process. If you omit this option, the new process uses COMINT's home terminal. For $<terminal-name>, specify a valid name for a terminal or process (that is, an alpha-numeric string of one to five characters, the first of which must be alphabetic).

<param-string>

is a program parameter or a string of parameters sent to the new process in the startup message. Leading blanks are deleted.

Considerations

- If you are not the SUPER.SUPER, you can debug only those programs whose process accessor IDs match your user ID. (For a description of process accessor IDs and process creator IDs, see the GUARDIAN Operating System User's Guide.)
- Privileged programs can be licensed (by the super ID) for use by users other than the super ID. However, only the super ID can debug privileged programs. If you are not the SUPER.SUPER and you try to use the RUND command to debug a licensed program, the program runs without entering a debug state.
- When you include the LIB option, the operating system tries to resolve external references to procedures in the program. First it searches the library file last specified with the LIB option when the program was run. (The date and time of the last modification, as well as the disc address of the library file, are stored in the program file.) When you run a program without specifying a library with the LIB option, the operating system compares the disc address and modification date of the actual library file with the information

about the library in the program file. If they do not match, the message "DIFFERENT LIBRARY CURRENTLY IN USE" is displayed. This safeguard prevents you from inadvertently running the wrong version of the library.

- The user who runs a program should have write access to the program file. For example, assume these two users:

  ```
  GROUPA.USER1 GROUPB.USER2
  ```

  and two files (with security CUCU) owned by USER1:

  ```
  $DATA.USER1.PROG
  $DATA.USER1.LIBFILE
  ```

  These two attempts to run program PROG by USER1 succeed:

  ```
  :RUN PROG
  :RUN PROG / LIB LIBFILE /
  ```

  USER2 (who does not have write access to PROG) can also run PROG, provided the library LIBFILE has not been altered since its last use. If LIBFILE has changed, however, and USER2 enters:

  ```
  :RUN PROG / LIB LIBFILE /
  ```

  the attempt to run the program fails, because USER2 does not have write access to PROG to enable external references to be resolved through the changed library file.

- When you give a RUN command and the new process begins executing, COMINT suspends itself unless your RUN command includes the NOWAIT option. If the new process does not take over break ownership, you can activate COMINT while the process executes by pressing the BREAK (or interrupt) key. COMINT then executes concurrently with the process. You can return COMINT to its suspended state with the PAUSE command.

- If you specify the NOWAIT option in your RUN command, COMINT returns to the command input mode as soon as the new process reads its startup message. Thus, NOWAIT means you do not have to wait for the new process to complete before you can enter other commands. NOWAIT is especially useful when you start up several programs using the IN <file-name> option.

- To run a process on a remote system, specify \<system-name> before the name of a program file. For example, this command runs an EDIT process on the \CHICAGO system:

  ```
  :\CHICAGO.EDIT
  ```

  Similarly, these commands also run an EDIT process on the \CHICAGO system, since the current default system name is used for file-name expansion:

  ```
  :SYSTEM \CHICAGO
  :EDIT
  ```

  A program file, however, must reside on the system where it is to run. Thus, the command:

  ```
  :RUN \DETROIT.MYPROG
  ```

  attempts to run a program file named:

  ```
  \DETROIT.<default-volume>.<default-subvolume>.MYPROG
  ```

  If no such program file exists on the remote system, the message "PROGRAM FILE ERROR 11" is displayed, indicating that the file does not exist. See the EXPAND Reference Manual for further information on creating remote processes.

Examples

This implicit RUN command runs the text editor program named EDIT that resides in the file $SYSTEM.SYSTEM.EDIT:

```
:EDIT
```

This command runs the program APP1 in the current default subvolume:

```
:RUN APP1
```

The following command:

```
:RUN APP1 / IN APP1IN, OUT APP1OUT, CPU 2, NAME $PROC1, &
:NOWAIT /
```

runs APP1 and also specifies:

- The file APP1IN as the input file for the program

- The file APP1OUT as the output file for the program
- Processor 2 as the processor where the program runs
- $PROC1 as the name of the process that is created
- That COMINT redisplay its command prompt without pausing for APP1 to complete execution

SET Command

Use the SET command to select the default debugger (DEBUG or INSPECT) that is to be in effect for processes started by COMINT and for descendants of those processes.

The syntax of the SET command is:

```
SET INSPECT { ON        }
            { OFF       }
            { SAVEABEND }
```

ON

selects the INSPECT symbolic debugger as the default debugger for all programs started by the current COMINT. (DEBUG is the system's default debugging utility.) INSPECT then prompts for input when any process created by the current COMINT (or by any descendant of the current COMINT) enters the debug state.

OFF

disables INSPECT and selects DEBUG as the default debugger. DEBUG then prompts for input when any process created by the current COMINT (or by any of its descendants) enters the debug state.

SAVEABEND

establishes INSPECT as the default debugger and automatically creates a save file if the program abends (terminates abnormally).

Considerations

- INSPECT is a symbolic debugger. It allows you to control running processes and SCREEN COBOL programs and to examine memory and modify data values--all with commands that use your source language. Both the NonStop systems and the NonStop 1+ system support INSPECT; INSPECT recognizes the same source-language commands for both systems.

  Besides the source-language commands, INSPECT supports machine-level commands for maximum debugging flexibility. (See the INSPECT Interactive Symbolic Debugger User's Guide for more information.) Also see the DEBUG, RUN, and SHOW commands in this section.

- Your selection of INSPECT as the default debugger is effective until you enter another SET command, or until you log off. After you log off, DEBUG once again becomes the default debugger. If, however, you enter the SET INSPECT command and later log on without first entering the LOGOFF command, then INSPECT remains the default debugger.

Example

You can specify INSPECT as the current default debugger by entering:

```
:SET INSPECT ON
```

You can see the results of the SET command by entering the SHOW command:

```
:SHOW
        INSPECT  ON
```

You can reinstate DEBUG as the default debugging utility by turning INSPECT off:

```
:SET INSPECT OFF
```

SETTIME Command

If you are logged on as SUPER.SUPER, you can use the SETTIME command to set the system's date and time-of-day clock. You normally use SETTIME after you cold load the first processor from disc, but before you load the rest of the system using the RELOAD command. You can also use SETTIME to reset the system clocks after a power failure (the interval clock in a processor module stops when power is interrupted).

The syntax of the SETTIME command is:

```
SETTIME  { <month-name> <day> }  <year>, <hour>:<min> [:<sec>]
         { <day> <month-name> }
         [ LCT ]
         [ LST ]
         [ GMT ]
```

<month-name>

is the name of the month. You must give at least the first three letters of the name of the month. You can use uppercase or lowercase for <month-name>.

<day>

is an integer in the range from 1 to 31, inclusive, designating the day of the month.

<year>

is the four-digit calendar year.

<hour>

is an integer in the range from 0 to 23, inclusive, indicating the hour of the day.

→

<min>

is an integer in the range from 0 to 59, inclusive, indicating the minute of the hour.

<sec>

is an optional integer in the range from 0 to 59, inclusive, indicating the seconds of the minute. If <sec> is omitted, 0 is assumed.

LCT, LST, or GMT

identifies the units of time. LCT is local civil time (that is, LST corrected for daylight savings time). LST is local standard time. GMT is Greenwich mean time. LCT is the default.

Consideration

If you execute this command during an XRAY measurement, invalid measurements will result.

Example

To set the system clock to 8:01 PM on March 5, 1985, enter:

```
:SETTIME MAR 5 1985, 20:01
```

You can see the new time by entering the TIME command:

```
:TIME
05 MAR 1985, 20:01
```

SHOW Command

Use the SHOW command to display the values of attributes set with the SET command. Currently, the only SET command attribute is INSPECT (specifying the current default debugger).

The syntax of the SHOW command is:

```
SHOW [ /OUT <list-file>/ ] [ INSPECT [ , <attribute> ... ] ]
```

SHOW

entered with no following parameters, displays all the attributes currently set with the SET command.

OUT <list-file>

is the name of a file to receive command output. By default, the output is sent to the OUT file in effect for the current COMINT (usually your terminal).

INSPECT

is the name of the alternate debugging utility--the INSPECT symbolic debugger. Also see the DEBUG, SET, and RUN commands in this section, and the INSPECT Interactive Symbolic Debugger User's Guide.

<attribute>

is any other attribute controlled by the SET command.

Example

To see whether INSPECT is the current default debugger, enter:

```
:SHOW
        INSPECT  OFF
```

STATUS Command

Use the STATUS command to display information about the state of running processes. Entering a STATUS command can give you the following information about processes:

- The program's file name, its process ID (its CPU number and process number) and its priority
- The process's home system and home terminal
- The process accessor ID of the process (for processes you start, this is normally your user ID)
- The cause of the wait state of any process that is waiting
- Whether the process contains privileged code, is waiting on a page fault, is on the ready list, or is any combination of these three factors
- The process's state
- The process's elapsed execution time

The syntax of the STATUS command is:

```
STATUS [ / OUT <list-file> / ] [ <range> ]
       [ , <condition> ] ...  [ , DETAIL ]
```

STATUS

entered with no <range> or <condition> parameters, reports the status of the last process you created that is still running, regardless of CPU. The display format for the STATUS command is shown in Figure 2-3.

OUT <list-file>

specifies a file to receive the STATUS output. If you omit the OUT option, the STATUS listing goes to the OUT file in effect for the current COMINT (usually the home terminal).

→

<range> is any one of these four:

```
[\<system-name>.]<cpu>,<pin>
[\<system-name>.]<cpu-number>
[\<system-name>.]$<process-name>
[\<system-name>.]*
```

- \<system-name> requests the status of all specified processes running in \<system-name>.
- <cpu>,<pin> requests the status of a particular process.
- <cpu-number> requests the status of all processes running in a particular CPU.
- $<process-name> requests the status of a particular named process or process pair.
- * requests the status of processes running in all CPUs.

If you do not specify <range>, STATUS reports on processes running at the current COMINT's primary CPU.

<condition> is any one of these four:

```
TERM [ $<terminal-name> ]
PRI [ <priority> ]
USER [ <ident> ]
PROG <program-file-name>
```

```
TERM [ $<terminal-name> ]
```

specifies processes running on a particular terminal. If you give no $<terminal-name>, STATUS reports on processes running at the current COMINT's home terminal.

→

PRI [<priority>]

specifies processes whose execution priority is less than or equal to the priority given. If you omit <priority>, STATUS reports on processes whose priority is one less than that of the current COMINT.

USER [<ident>]

specifies processes created by a particular user, where <ident> is either <group-name>.<user-name> or <group-id>,<user-id>. If you include USER without <ident>, STATUS reports on processes whose process accessor ID matches your user ID.

PROG <program-file-name>

specifies processes with the given program file name.

If you specify more than one <condition>, STATUS reports on all processes that satisfy all of the conditions.

DETAIL

gives a detailed display of process status. Display format for the DETAIL option is shown in Figure 2-4.

Listing Formats

The STATUS command uses the format shown in Figure 2-3 to display the status of processes in a processor module.

```
                    SYSTEM \<system-name>

PID   PRI   PFR  %WT  USERID  MYTERM  PROGRAM FILE NAME

<cp> <pri> <pfr> <wf>  <g,u>  <term>    <prog-name>
      [ LIBRARY FILE NAME:       <library-file-name> ]
      [ SWAP    FILE NAME:          <swap-file-name> ]
```

Figure 2-3. STATUS Command Listing Format

The variables that appear in Figure 2-3 are:

\<system-name>	is the name of the system where the listed processes are running.
<cp>	is the process ID (CPU and process number) of the process being reported.
<pri>	is the process's execution priority.
<pfr>	P, F, and/or R appear here: P indicates that the process contains privileged code; F indicates the process is waiting on a page fault; R indicates the process is on the ready list.
<wf>	is the wait field, which indicates whether the process is waiting and gives the cause of the wait. The <wf> value is obtained from the wait field of the awake-wait word in the process control block (PCB) for the process. The wait field in the PCB can have the following values:

```
wait-field.<8>  wait on PON   -- CPU power on
          .<9>  wait on IOPON -- I/O power on
          .<10> wait on INTR  -- interrupt
          .<11> wait on LINSP -- INSPECT event
          .<12> wait on LCAN  -- message system, cancel
```

.<13> wait on LDONE -- message system, done
.<14> wait on LTMF -- TMF request
.<15> wait on LREQ -- message system, request

The bits in the wait field are numbered from left to right; thus, octal 3 (%003) means that bits 14 and 15 are set. Typical values for <wf> and their meanings are:

Value	Meaning
%000	Process is running; or process was waiting on an event that has since occurred and is now ready to run; or process is in call to DELAY; or process is suspended.
%001	Process is waiting for a message to occur on its $RECEIVE file.
%002	Process is waiting for TMF request to complete, or user process is waiting for ENDTRANSACTION to complete.
%004	Process is waiting for input/output or interprocess request to complete.
%005	Process is waiting for call to AWAITIO for I/O completion on any file.

<g,u> is the <group-id>,<user-id> of the process accessor ID for the process.

<term> is the process's home terminal. If this terminal is connected to a remote system in a network, the system name precedes the terminal name (such as \CHICAGO.$TERM49).

<prog-name> is the name of the program file. For system processes, <prog-name> is $SYSTEM.SYS<nn>.OSIMAGE. (Subvolume $SYSTEM.SYS<nn> contains the GUARDIAN operating system image currently in use; <nn> is a two-digit octal integer which identifies that subvolume.)

<library-file-name> is the name of the user library file. This line appears only when you request the status of a single process (or a process pair) that is running with a user library (such as one specified with the LIB option of the RUN command).

If you specify the DETAIL option, the STATUS command listing format is as shown in Figure 2-4. The additional information listed by the DETAIL option includes extended memory swap file names, process time, and process state.

```
SYSTEM: \<system-name>                    <date>  <time>
PID: <cp> [ ($<proc-name>) ]
PRIORITY: <pri>
[ PRIV ] [, PAGE FAULT] [ , READY ]
WAIT STATE: <wf> [ <wait-state-name> ]
USERID: <g,u> (<user-name>)
MYTERM: $<term>
PROGRAM FILE NAME: <prog-name>
[ LIBRARY FILE NAME: <library-file-name> ]
PROCESS TIME: <elapsed-exec-time>
PROCESS STATES: <process-state>
```

Figure 2-4. STATUS Command Listing Format (DETAIL Option)

Variables that appear in both Figure 2-3 and Figure 2-4 are defined following Figure 2-3. Variables that appear only in Figure 2-4 are:

<date> <time>	are the current system date (day, month, and year) and time.
PRIV, PAGE FAULT, READY	PRIV indicates that the process contains privileged code; PAGE FAULT indicates the process is waiting on a page fault; READY indicates the process is on the ready list.
<wait-state-name>	is the name of the event that the process is waiting on. These event names are listed in the description of the <wf> variable following Figure 2-3.
<user-name>	is the group name and user name of the user ID that matches this process's process accessor ID.
<elapsed-exec-time>	is the process time that has elapsed.

<process-state> gives the current process state. Any of the following states may be listed:

```
[TEMPORARY] [,LOGGED ON] [,PENDING]
{ STARTING | RUNNABLE | SUSPENDED |
DEBUG MAB | DEBUG BREAKPOINT |
DEBUG TRAP | DEBUG REQUEST | INSPECT MAB |
INSPECT BREAKPOINT | INSPECT TRAP | INSPECT
REQUEST | SAVE ABEND | TERMINATING }
```

Examples

You can view the status of all processes started from your terminal by entering:

```
:STATUS *, TERM
```

```
                                    SYSTEM  \MANUF
PID     PRI PFR %WT USERID    MYTERM        PROGRAM FILE NAME

04,054 150 P R 000 008,044    $FRED     $SYSTEM .SYS02    .COMINT

05,009 150 P   001 008,044    $FRED     $SYSTEM .SYS02    .COMINT

08,043 148     000 008,044    $FRED     $SYSTEM .SYS02    .EDIT

08,050 148     001 008,044    $FRED     $SYSTEM .SYS02    .VS
```

You can view the status of all processes started by user MANUF.FRED by entering:

```
:STATUS *, USER MANUF.FRED
```

You can view the status of process $MYPR by entering:

```
:STATUS $MYPR
```

Including the DETAIL parameter in a STATUS command yields a display such as this:

```
:STATUS 1, 20, DETAIL
SYSTEM: \TS                    16JUL84 13:25
PID: 01,020  ($C33)
PRIORITY: 151
WAIT STATE: %005 (LDONE, LREQ)
USERID: 001,094 (SOFTWARE.DAVE)
MYTERM: $TERM1
PROGRAM FILE NAME: $SYSTEM.SYS05.COMINT
SWAP FILE NAME: $SYSTEM.#1234
PROCESS TIME: 00:01:12.345
PROCESS STATES: RUNNABLE
```

STOP Command

You can use the STOP command to delete a process (stop a running program).

The syntax of the STOP command is:

```
STOP [ [ \<system-name>.] { $<process-name> }
                          { <cpu>,<pin>     } ]
```

STOP

entered with no following parameters, stops the latest process you started from the current COMINT.

\<system-name>

is the name of the system running the process you want to stop. If the process is running on your current default system, you can omit \<system-name>.

$<process-name>

is the name of a named process that you want to stop.

<cpu>,<pin>

is the process ID (CPU number and process number within that CPU) of the process you want to stop.

Considerations

- Standard users can stop only the processes with creator accessor IDs that match their user IDs. (Refer to the EXPAND Reference Manual for remote process restrictions.)
- SUPER.SUPER can stop any user process.

- A group manager can stop any process whose creator accessor ID matches any user ID in the group.

Examples

To stop the last process you started from the current COMINT, enter:

```
:STOP
```

If you started the process whose process ID is 0,18, or if you are the super ID (SUPER.SUPER), you can stop the process by entering:

```
:STOP 0,18
```

If you are a group manager and a member of your group started the process named $END, or if you are the super ID, you can stop the process $END by entering:

```
:STOP $END
```

SUSPEND Command

Use the SUSPEND command to temporarily suspend a process (or process pair) in order to prevent it from competing for system resources. A suspended process or process pair cannot execute instructions until you reactivate it. To reactivate a suspended process or process pair, use the ACTIVATE command. (Also, another process can call the ACTIVATEPROCESS procedure to reactivate the suspended process or process pair.)

The syntax of the SUSPEND command is:

```
SUSPEND [ [ \<system-name>.] { $<process-name> }  ]
                             { <cpu>,<pin>     }
```

SUSPEND

entered with no following parameters, suspends the latest process you started from the current COMINT.

\<system-name>

is the name of the system running the process you want to suspend. If the process is running on your current default system, you can omit \<system-name>.

$<process-name>

is the name of a named process that you want to suspend.

<cpu>,<pin>

is the process ID (CPU number and process number within that CPU) of the process you want to suspend.

Considerations

- COMINT does not suspend the process (or process pair) you name in your SUSPEND command until that process is ready to execute instructions. Therefore, any outstanding waits (such as for I/O completion) are satisfied before the process or process pair is suspended.
- Standard users can suspend only processes that they started (and descendants of those processes). That is, the current user's process accessor ID must match the process accessor ID of the process to be suspended. (See the EXPAND Reference Manual for restrictions on remote processes.)
- The super ID user (SUPER.SUPER) can suspend any process.
- A group manager can suspend any process whose process accessor ID matches any user ID in the group.

Examples

To suspend the last process you started from the current COMINT, enter:

```
:SUSPEND
```

If you started a process with process ID 0,18, or if you are the super ID, you can suspend the process by entering:

```
:SUSPEND 0,18
```

If you are a group manager and a member of your group started the process whose name is $CLOCK, or if you are the super ID, you can suspend the process by entering:

```
:SUSPEND $CLOCK
```

SWITCH Command

Use the SWITCH command to switch the functions of a COMINT process pair. The backup process becomes the primary process, and vice versa.

The syntax of the SWITCH command is:

```
SWITCH
```

Example

Suppose that the primary COMINT process which controls your terminal is executing in CPU 5, and the backup COMINT process is executing in CPU 4. (To display the CPU numbers of the processors where your COMINT processes are running, use the WHO command.)

You can switch the functions of these processes (make the COMINT process executing in CPU 4 the primary process and the process executing in CPU 5 the backup) by entering the SWITCH command:

```
:SWITCH
```

SYSTEM Command

You can use the SYSTEM command to set a new current default system. This command is valid only for systems that are named (such as systems within a network).

The syntax of the SYSTEM command is:

```
SYSTEM [ \<system-name> ]
```

SYSTEM

entered with no following parameters, establishes as the new current default system the system where the current COMINT is running.

\<system-name>

is the name of a new current default system.

Considerations

- The system you specify in a SYSTEM command is temporarily in effect. After you enter a LOGON command, or a SYSTEM or VOLUME command with no following parameters, your default system is again your logon default. (To change your logon defaults, use the DEFAULT program).
- If you are running a remote COMINT process, entering SYSTEM with no following parameters establishes that remote system (not the local system to which your terminal is connected) as your current default system.

- These commands are not equivalent:

  ```
  :SYSTEM \<local-system-name>

  :SYSTEM
  ```

  The first command causes the network restrictions on file-name lengths to take effect; the second does not. See the EXPAND Reference Manual for information on network file-name restrictions.

- If the current default volume name contains seven characters after the dollar sign ($), you cannot specify a new current default system name with the SYSTEM command.

Examples

MANUF.FRED specifies \LONDON as the current default system by entering:

```
:SYSTEM \LONDON
```

When Fred enters the WHO command, the current default system is shown:

```
HOME TERMINAL: $FRED
COMMAND INTERPRETER: \SFO.$C183  PRIMARY CPU: 04  BACKUP CPU: 05
CURRENT VOLUME: $BIG.JOBS   CURRENT SYSTEM: \LONDON
USERID: 008,044   USERNAME: MANUF.FRED   SECURITY: NUNU
```

Note that the home system of Fred's COMINT process has not changed; COMINT is still running on \SFO. Also, the current default volume and subvolume names are not changed by the SYSTEM command. However, the \LONDON system may contain no files in a subvolume by the name $BIG.JOBS, which is Fred's default subvolume on \SFO.

Fred can restore his logon default system (the one in effect when he logged on or last entered the DEFAULT command) by entering:

```
:SYSTEM
```

SYSTIMES Command

Use the SYSTIMES command to display the current date and time (in local civil time and Greenwich mean time) and also the date and time when the system was last cold loaded.

The syntax of the SYSTIMES command is:

```
SYSTIMES
```

Listing Format

The SYSTIMES command displays three lines of information giving you the date and time as shown in Figure 2-5.

```
<dd><mmm><yyyy>, <hh>:<mm>:<ss>.<mmmuuu>  LCT
<dd><mmm><yyyy>, <hh>:<mm>:<ss>.<mmmuuu>  GMT
<dd><mmm><yyyy>, <hh>:<mm>:<ss>.<mmmuuu>  COLD LOAD (LCT)
```

Figure 2-5. SYSTIMES Command Listing Format

The variables that appear in Figure 2-5 are:

<dd>	is the day of the month (01, 02, ... , 31).
<mmm>	is a three-letter abbreviation for the ASCII month of the year (JAN, FEB, ... , DEC).
<yyyy>	is the year (1984, 1985, ...).
<hh>	is the hour of the day (00, 01, ... , 23).
<mm>	is the minutes of the hour (00, 01, ... , 59).

<ss> is the seconds of the minutes (00, 01, ... , 59).

<mmmuuu> is the millisecond and microsecond of the second (000000, 000001, ... , 999999).

LCT appears on the line that shows the current date and time in local civil time.

GMT appears on the line that shows the current date and time in Greenwich mean time.

COLD LOAD (LCT)
appears on the line that shows the date and time of the most recent cold load. This time is given in local civil time.

Consideration

The SYSTIMES command applies only to the local system; to obtain the times from a remote system, you must first start a COMINT process there and then enter the SYSTIMES command.

Example

To display the various system times, enter:

```
:SYSTIMES
04 OCT 1985, 10:02:39.337905 LCT
04 OCT 1985, 17:02:39.337905 GMT
03 OCT 1985, 09:25:30.275748 COLD LOAD (LCT)
```

TIME Command

Use the TIME command to display the current setting of the system's date and time-of-day clock.

The syntax of the TIME command is:

```
TIME
```

Listing Format

The TIME command displays the current date and time as shown in Figure 2-6.

```
<dd> <mmm> <yyyy>, <hh:mm>
```

Figure 2-6. TIME Command Listing Format

The variables that appear in Figure 2-6 are:

<dd>	is the day of the month (01, 02, ... , 31).
<mmm>	is the three-letter abbreviation for the ASCII month of the year (JAN, FEB, ... , DEC).
<yyyy>	is the year (1984, 1985, ...).
<hh>	is the hour of the day (00, 01, ... , 23).
<mm>	is the minutes of the hour (00, 01, ... , 59).

Consideration

The TIME command applies only to the local system; to obtain the time from a remote system, run a COMINT process there and then enter the TIME command.

Example

You can display the current date and time by entering:

```
:TIME
05 MAY 1985, 14:45
```

USERS Program

The USERS program lists user attributes for a particular user or range of users.

The syntax of the command to start a USERS process is:

```
USERS [ / OUT <list-file> / ] [ <range> ]
```

NOTE

A command to start a USERS process can include any run option for the RUN command.

OUT <list-file>

names a file to receive the USERS listing. By default, the listing is sent to the current COMINT's OUT file, which is usually the home terminal.

<range>

specifies a particular user or group of users to be listed. Note that <range> refers to the local system only. The allowable range specifications and their meanings are:

<range> Specification	Meaning
(blank)	Lists entry for current user only.
<group-id>,<user-id>	Lists entry for user with specified user number.
<group-id>, *	Lists entries for all users in the specified group.
<group-name>.<user-name>	Lists entry for the named user.

→

[<group-name>.]*	Lists entries for all users in the named group. If you omit <group-name>, USERS lists all users in the group of the logged-on user.
. or *,*	Lists entries for all users.

Listing Format

The USERS program lists information in the format shown in Figure 2-7.

```
GROUP . USER      I.D. #    SECURITY    DEFAULT VOLUMEID

<group>.<user>     <g,u>      <rwep>     $<vol>.<subvol>
```

Figure 2-7. USERS Program Listing Format

The variables that appear in Figure 2-7 are:

<group>.<user>	are the group and user names of the listed user.
<g,u>	are the <group-id> and <user-id> of the listed user.
<rwep>	is the logon default security for the listed user.
$<vol>.<subvol>	are the logon default volume and subvolume for the listed user.

Example

To display the USERS listing for yourself (the currently logged on user), enter:

```
:USERS
```

To find out the user name associated with the user ID 8,44, enter:

```
:USERS 8,44
```

The USERS program displays information such as this:

```
GROUP . USER          I.D. #    SECURITY    DEFAULT VOLUMEID

MANUF    .FRED        008,044      NUNU      $BIG.BAD
```

You can get this information for all users in the PARTS group by entering:

```
:USERS PARTS.*

GROUP      USER         I.D. #    SECURITY    DEFAULT VOLUMEID

PARTS     .CLYDE        001,000     GGGO     $SYSTEM.ENGINE
PARTS     .MARY         001,002     AOGO     $SYSTEM.WHEELS
PARTS     .JOE          001,001     CUCU     $SYSTEM.TRANS
PARTS     .MANAGER      001,255     GGGA     $SYSTEM.PAYROLL
```

VOLUME Command

Use the VOLUME command to temporarily change your current settings for default volume, subvolume, and security.

The syntax of the VOLUME command is:

```
VOLUME [ <default-names> ] [ , "<default-file-security>" ]

<default-names> is one of the following:

   $<default-volume-name>

      is a new current default volume name; the current
      default subvolume is unchanged.


   <default-subvol-name>

      is a new current default subvolume name; the current
      default volume name is unchanged.


   $<default-volume-name>.<default-subvol-name>

      are new current defaults for both volume and subvolume
      names.


<default-file-security>

   sets your current default disc file security.  The
   system assigns this file security to newly created disc
   files unless you explicitly assign a different security
   when you create the file.  Specify <default-file-
   security> as a four-character string "<rwep>" to specify
   the security for each type of file access:  read, write,
   execute, and purge.  For <rwep> you can specify A, G, O,
   N, C, or U, as described in Table 2-1.  Also see the
   DEFAULT command in this section and the File Utility
   Program SECURE command in Section 3.
```

Considerations

- Settings made with the VOLUME command are only temporarily in effect--until you enter a LOGON command or a SYSTEM or VOLUME command with no following parameters. For example, if you log on again, all your current defaults are reset to the original logon default system, volume, subvolume, and security (or as you specified in your last DEFAULT command).
- If the current default system is a remote system, you cannot specify a new current default volume name that has seven characters after the dollar sign ($).
- Likewise, you cannot specify a new default system with the SYSTEM command if the current default volume name contains seven characters after the dollar sign ($).
- Entering VOLUME with no parameters resets your current default system, volume, subvolume, and security to the values in effect at logon, or as they were set with your last DEFAULT command. Note, however, that you cannot explicitly change your current default system with the VOLUME command.
- To display your logon default security and volume and subvolume names, enter the USERS command.

Examples

To change your current default subvolume to $MANUF.BILLS, enter:

```
:VOLUME $MANUF.BILLS
```

You can change the default security given to the files you create to "CUCU" by entering:

```
:VOLUME , "CUCU"
```

You can reestablish the default system, volume, subvolume, and security that were in effect when you logged on (unless you subsequently changed your logon default setting with the DEFAULT program) by entering:

```
:VOLUME
```

WAKEUP Command

You can use the WAKEUP command to set COMINT's wakeup mode.

The syntax of the WAKEUP command is:

```
WAKEUP  { ON | OFF }
```

ON

means that COMINT is awakened (that is, returned from the paused state) when any process you started with a RUN command is deleted.

OFF (the logon default setting)

means that COMINT is awakened only when the latest process you started with a RUN command is deleted, or when a process you designate in a PAUSE command is deleted.

Example

To wake up COMINT and return control of the terminal to COMINT whenever a process that you started is deleted (by entering an explicit or implicit RUN command from this COMINT process), enter:

```
:WAKEUP ON
```

WHO Command

Use the WHO command to display your current default settings and other information relating to your current COMINT process.

The syntax of the WHO command is:

```
WHO [ / OUT <list-file> / ]

OUT <list-file>

   names a file to receive the output from the WHO command.
   If you omit this parameter, the output is displayed on
   the OUT file in effect for the current COMINT (usually
   the home terminal).
```

Listing Format

The WHO command displays information about your current default settings in the format shown in Figure 2-8.

```
HOME TERMINAL: $<terminal-name>
COMMAND INTERPRETER:<CI> [PRIMARY CPU:<p> BACKUP CPU:<b>]
CURRENT VOLUME:<defaults> [CURRENT SYSTEM:\<default-sys>]
USERID: <g,u> USERNAME: <group>.<user> SECURITY: <rwep>
```

Figure 2-8. WHO Command Listing Format

The variables that appear in Figure 2-8 are:

$<terminal-name>	is the name of the current COMINT's home terminal.
<CI>	is the process name of the current COMINT. If your system is part of a network or is named, <CI> includes the system name (see the example). If the COMINT process is not named, its process ID (cpu>,<pin>) is displayed.
<p> and <b>	are the CPU numbers of current COMINT's primary and backup processes. If the primary COMINT process is named but has no backup, the CPU number for the backup does not appear. If the primary is not named, neither CPU number appears (in this case, the primary COMINT process's CPU number is shown in <CI>).
<defaults>	are your current default volume and subvolume.
\<default-sys>	is your current default system. The CURRENT SYSTEM field does not appear in the display if the current system is the one where the COMINT process is running, and you have not specified a new current default system with the SYSTEM command (see "Considerations").
<g,u>	are your group ID and user ID.
<group>.<user>	are your group name and user name.
<rwep>	is the current security to be assigned to files you create.

Considerations

- The WHO command is useful if you forget your current defaults. For example, if you are on a network and use the SYSTEM command to access another system, you can issue a WHO command to check your defaults before you run programs or purge files. Checking your defaults helps you avoid errors such as running programs in the wrong system, or purging remote files instead of local files (or vice versa).

- The WHO command displays the name of the current default system only if your COMINT process is not running on your logon default system, or if you specified a current default system with the SYSTEM command. This fact can be a possible source of confusion if your COMINT process is not running on the system to which your terminal is connected.

  For example, suppose your terminal is connected to the system \LOCAL, but your COMINT process is running on the system \REMOTE. The current system does not appear in the WHO display if you issue the following commands:

  ```
  :SYSTEM
  :WHO
  ```

  Nevertheless, \LOCAL is not the current default system. The <CI> field displays the current default system, \REMOTE.

Example

User MANUF.FRED logs on to system \ROME and then uses the SYSTEM command to make \NAPLES his current default system. When he enters a WHO command, this information is displayed:

```
HOME TERMINAL: $FRED
COMMAND INTERPRETER: \ROME.$C183  PRIMARY CPU: 04  BACKUP CPU: 05
CURRENT VOLUME: $MORE.DATA   CURRENT SYSTEM: \NAPLES
USERID: 008,044   USERNAME: MANUF.FRED   SECURITY: NUNU
```

{X|Y}BUSDOWN Command

If you are a system operator (with group ID 255) or are logged on as SUPER.SUPER, you can use the XBUSDOWN or YBUSDOWN command to inform the operating system that an interprocessor bus is being downed and should not be used. This command is equivalent to the BUSCMD process.

The syntax of the {X|Y}BUSDOWN command is:

```
{ X | Y }BUSDOWN <from-cpu>, <to-cpu>
```

{X|Y}

specifies the bus (X or Y) to be brought down.

<from-cpu>, <to-cpu>

are CPU numbers from 0 to 15, inclusive. Each CPU number specifies one endpoint of a segment of the designated bus. On this segment, transfers are not permitted. Specify -1 to indicate all processors.

Example

In a four-processor system, SUPER.SUPER can bring down the X bus from processor 1 to all other processors by entering:

```
:XBUSDOWN 1, -1
THE X BUS FROM CPU 01 TO 00 HAS BEEN DOWNED.
THE X BUS FROM CPU 01 TO 01 HAS BEEN DOWNED.
THE X BUS FROM CPU 01 TO 02 HAS BEEN DOWNED.
THE X BUS FROM CPU 01 TO 03 HAS BEEN DOWNED.
```

{X|Y}BUSUP Command

If you are a system operator (with group ID 255) or are logged on as SUPER.SUPER, you can use the XBUSUP or YBUSUP command to inform the operating system that an interprocessor bus is available for use. This command is equivalent to the BUSCMD process.

The syntax of the {X|Y}BUSUP command is:

```
{ X | Y }BUSUP <from-cpu>, <to-cpu>
```

{X|Y}

specifies the bus (X or Y) to be brought up.

<from-cpu> and <to-cpu>

are CPU numbers, from 0 to 15, inclusive. Each CPU number specifies the endpoint of a segment of the designated bus. On this segment, transfers are now permitted. Specify -1 to indicate all processors.

Example

SUPER.SUPER can bring the Y bus back up from processor 1 to processor 2 by entering:

```
:YBUSUP 1,2
THE Y BUS FROM CPU 01 TO 02 HAS BEEN UPPED.
```

SECTION 3

THE FILE UTILITY PROGRAM (FUP)

With the File Utility Program (FUP), you can perform many functions involving both disc files and devices such as tape drives. Use FUP to create and purge files, duplicate and display files, alter file characteristics, and load data into files.

This section begins with a description of how to run FUP and a summary of the commands you can use in FUP. Next is a description of important syntax conventions used in this section. The major portion of this section is the syntax descriptions of FUP commands, arranged in alphabetical order. You should read and understand the introductory subsections before using the FUP command syntax descriptions.

Useful background reading includes the GUARDIAN Operating System User's Guide, which describes the basic uses of FUP. You should also be familiar with the types of disc files described in that manual and in the ENSCRIBE Programming Manual.

RUNNING THE FUP PROGRAM

FUP normally resides in a file named $SYSTEM.SYS<nn>.FUP. $SYSTEM.SYS<nn> is the subvolume containing the GUARDIAN operating system image currently in use, and <nn> is a two-digit octal integer which identifies that subvolume.

You access FUP from the GUARDIAN operating system command interpreter (COMINT). You can enter FUP commands in either of two ways:

- Through COMINT, by typing FUP followed by a FUP command and a RETURN at COMINT's prompt, a colon (:)
- Through FUP, by typing FUP and a RETURN to start a FUP process and then entering FUP commands at FUP's prompt, a hyphen (-). (FUP's one-line sign-on banner appears before it displays its prompt.)

The syntax of the command to run FUP is:

```
FUP [ / <run-option> [ , <run-option> ]... / ] [ <command> ]
```

FUP

is an implicit RUN command instructing COMINT to run the program located in $SYSTEM.SYS<nn>.FUP.

- If you type FUP followed by a <command>, you initiate a FUP process that executes the command and then terminates, returning control of the terminal to the command interpreter.
- If you type FUP with no following <command>, you initiate a FUP process that takes control of the terminal and displays its prompt, a hyphen (-). You can then enter FUP commands at the prompt. To exit FUP, type the EXIT command.

<run-option>

can be any <run-option> valid for the RUN command in COMINT. For a complete list of run options, see the description of the RUN command in the COMINT section. Two options used most often with FUP are:

```
IN <file-name>
OUT <list-file>
```

→

IN <file-name>

names a disc file, a nondisc device, or a process from which FUP is to read commands. FUP reads 132-byte records from <file-name> until it reaches an end-of-file marker. Only one command is permitted for each record.

If you omit this option, FUP uses the IN <file-name> that is in effect for the current COMINT process. If you are using COMINT interactively, as is usual, the home terminal is the IN <file-name> for the current COMINT process.

OUT <list-file>

names a nondisc device, a process, or an existing disc file to which FUP is to direct its listing output (unless a command parameter directs output elsewhere.) Partial file names are expanded using the current default names for system, volume, and subvolume, as necessary.

If <list-file> is an unstructured disc file, each list file record is 132 characters; partial lines are blank-filled through column 132.

If you omit this option, FUP uses the OUT <list-file> that is in effect for the current COMINT process. If you are using COMINT interactively, as is usual, the home terminal is the OUT <list-file> for the current COMINT process.

<command>

is a FUP command, limited to 132 characters. To continue a long command line beyond the screen width, end the line with an ampersand (&) followed by a RETURN; COMINT redisplays its command prompt on the next line, and you can continue typing your command. FUP executes your <command> and then terminates. Descriptions of all FUP commands are given in this section.

Considerations

- The number of FUP commands you plan to enter can influence your choice of method for entering FUP commands. If you have a large number of FUP operations to perform, you can save time by starting a FUP process and entering your commmands at FUP's prompt. This method is faster than entering a series of single FUP commands at COMINT's prompt. On the other hand, if you have only one or two FUP commands to enter, you can save time by entering the complete command at COMINT's prompt rather than starting a FUP process, entering a command, and then typing EXIT.

- You can use the FC (Fix Command) command in FUP just as you do in COMINT. (For details on usage, see the description of the FC command in the COMINT section of this manual).

Examples

The following FUP command displays file information for all the files in the current subvolume:

```
:FUP INFO *
```

This command line tells FUP to read input from the disc file OBEY1 and to send the output of the FUP commands in OBEY1 to the disc file LIST1:

```
:FUP / IN OBEY1, OUT LIST1 /
```

The following example demonstrates starting a FUP process and entering FUP commands interactively through FUP. These command lines (1) start a FUP process, (2) duplicate the file named FIRST to the file named SECOND, and (3) purge the original contents of SECOND. (Both files are in the default subvolume and volume.) After FUP completes the DUP operation, it redisplays its hyphen prompt:

```
:FUP
-DUP FIRST, SECOND, PURGE
-
```

FUP COMMAND SUMMARY

FUP commands, grouped according to function, are listed and described here and on the following pages.

The commands to control FUP are:

BREAK key	Aborts the currently running FILES, INFO, SUBVOLS, COPY, or PURGE (prompting form) command. If no FUP command is being executed or if the current command is not a listing function, control of the terminal returns to COMINT. The FUP process continues to execute even after COMINT regains control.
<	Means that the text which follows on the same line is a comment. Use the less-than symbol to introduce comment lines in FUP command files.
ALLOW	Sets the number of errors and warnings allowed during the execution of FUP commands.
End of file (EOF)	Terminates the FUP process and returns control of the terminal to COMINT. EOF is the same as typing EXIT; it can take the form of the EOF of an IN file or of typing CTRL/Y on the terminal.
EXIT	Terminates the FUP process.
FC	(Fix Command) Allows you to edit and/or reexecute a command line. (For details, see the FC command in the COMINT section.) This command is for interactive use only.
HELP	Lists the names of all FUP commands or displays the syntax of a particular command.
SYSTEM	Sets the current default system name used by FUP.
VOLUME	Sets the current default disc volume and subvolume names used by FUP.

The commands to display information are:

FILES	Displays the names of files in a given subvolume or volume.
INFO	Displays file characteristics of one or more files.
LISTOPENS	Lists all processes that currently have open one or more designated files.
SUBVOLS	Displays the names of all subvolumes on a designated volume.

The commands to create files are:

CREATE	Creates a file using the current file-creation parameter values.
RESET	Changes one or more file-creation parameter values to the default settings.
SET	Sets one or more file-creation parameter values for subsequent file creations.
SHOW	Displays the current settings of the file-creation parameter values.

The commands to copy files and display the contents of files are:

COPY	Makes a record-by-record copy between files on the same or different media. Can also display the contents of a file.
DUP[LICATE]	Makes a copy of one or more disc files.

The commands to allocate and deallocate file extents are:

ALLOCATE	Preallocates a specified number of file extents for one or more disc files.
DEALLOCATE	Deallocates any extents past the end-of-file extent for one or more disc files.

The commands to rename and purge files are:

PURGE	Purges one or more disc files.
PURGEDATA	Purges data from one or more disc files.
RENAME	Renames one or more disc files.

The commands to set file security are:

GIVE	Changes a file owner ID for one or more files.
LICENSE	(super ID command) "Licenses" one or more program files containing privileged procedures so nonprivileged users can run the files.
REVOKE	(super ID command) Revokes the "license" of one or more program files to execute with privileged procedures.
SECURE	Sets file security attributes for one or more disc files.

The command to alter file characteristics is:

ALTER	Changes selected characteristics of a disc file.

The commands to load files are:

BUILDKEYRECORDS	Writes the alternate-key records for key fields of a specified structured disc file to a destination file (usually a magnetic tape). You can then load the alternate-key records from the destination file into the alternate-key file with the COPY or LOAD commands.
LOAD	Loads data into a structured disc file without affecting any associated alternate-key files. For key-sequenced files, the input data can be unsorted or sorted. (Unless you specify sorted, the LOAD command assumes data is unsorted and performs a sort of the input records before loading the file.) For key-sequenced files, you can also specify slack space for future insertions to the file.
LOADALTFILE	Loads an alternate-key file with the alternate-key records of a specified structured disc file. You can specify slack space for future insertions.

The command to check file integrity is:

CHECKSUM	Recomputes the checksum value for each block of data in a file.

FUP SYNTAX CONVENTIONS

The syntax descriptions of FUP commands in this section use the syntax conventions presented at the front of this manual. You should read and understand these conventions before you work with the syntax descriptions.

In addition, three standard variables appear repeatedly in this section. In using syntax descriptions, you need to know these variables:

- <list-file>
- <fileset>
- <fileset-list>

This subsection describes how to specify the parts of FUP commands that are symbolized by these syntax variables.

Specifying <list-file>

The <list-file> specification designates a nondisc device, a process, or an existing disc file where you want to direct the listing output of a command. Partial file names are expanded using the current default names for system, volume, and subvolume, as necessary. In syntax descriptions, <list-file> usually appears with the OUT keyword.

If you specify an unstructured disc file, each list-file record is 132 characters (partial lines are blank-filled through column 132). If you give no <list-file>, FUP uses the OUT <list-file> that is in effect for the current FUP process. In interactive use, the OUT <list-file> for FUP is normally the home terminal.

Specifying <fileset>

The <fileset> specification designates a set of related files. A <fileset> can be one file, all the files in a subvolume, or all the files on a volume.

The form of <fileset> is:

```
[ \<system-name>.]<fileset-spec>

\<system-name>

   is the name of a system in a network.  If you omit
   \<system-name>, FUP uses the current default system--
   that is, the system that was the current default when
   you started FUP, or the system you set in your last
   FUP SYSTEM or VOLUME command.

<fileset-spec> is:

   [ $<volume-name>.][<subvolume-spec>.]<disc-file-spec>

   $<volume-name>

      is the name of a volume (disc drive).  If you omit
      $<volume-name>, FUP assumes the current default
      volume--that is, the volume that was the current
      default when you started FUP, or the volume you set
      in your last FUP VOLUME command.

   <subvolume-spec> is either <subvolume-name> or *

      <subvolume-name> is the name of a subvolume.

      * (the asterisk) means all subvolumes on the
      designated volume.  If you use * for
      <subvolume-spec>, then you must also give
      <disc-file-spec> as *.
```

→

If you omit <subvolume-spec>, FUP assumes the current default subvolume--that is, the subvolume that was the current default when you started FUP, or the subvolume set in your last FUP VOLUME command.

<disc-file-spec> is either <disc-file-name> or *

<disc-file-name> is the name of a disc file.

* means all files in the subvolume named in <subvolume-spec>.

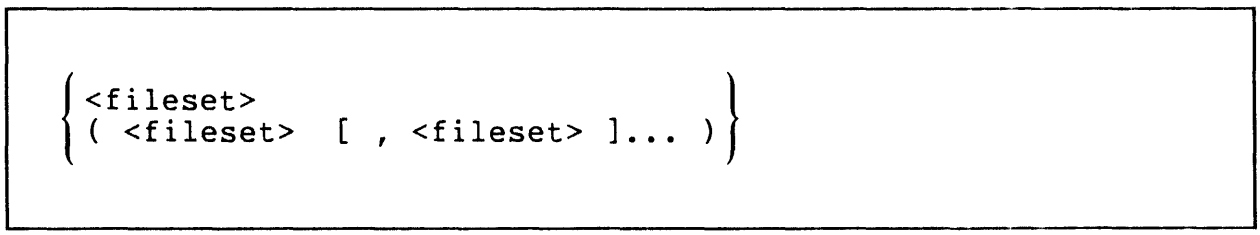
Some examples of <fileset> follow:

MYFILE	refers to the file:
$<default-volume-name>.<default-subvolume-name>.MYFILE	
MYSVOL.*	refers to all the files in the subvolume MYSVOL on the current default volume.
$VOL1.*	refers to all the files in the current default subvolume on the volume $VOL1.
.	refers to all files on the current default volume.
\NY.$BRDWAY.MYTOWN.*	refers to all files in the subvolume MYTOWN on the volume $BRDWAY on the \NY system.

Specifying <fileset-list>

The <fileset-list> specification designates one or more sets of related files. A <fileset-list> consists of one <fileset> or more, up to a maximum of 10 <fileset>s.

The form of <fileset-list> is:



Some examples of <fileset-list> follow:

(MYFILE, MYSRC, MYOBJ)	refers to the files MYFILE, MYSRC, and MYOBJ in the current default subvolume on the current default volume.
(*, $VOL1.*)	refers to all files in the current default subvolume on both the current default volume and the volume $VOL1.
(*.*, $VOL2.*.*)	refers to all files on the current default volume and all files on the volume $VOL2.

FUP COMMAND DESCRIPTIONS

The rest of this section contains descriptions of the syntax for FUP commands. Each command description contains:

- A summary of the command's function
- The syntax of the command, including a description of the syntax parameters and variables
- The listing format used for command output if the command produces a display or listing output
- Considerations for the use of the command
- Examples of command usage

ALLOCATE Command

Use the ALLOCATE command to preallocate file extents for a disc file.

The syntax of the ALLOCATE command is:

```
ALLOCATE <fileset-list> , <num-extents> [ , PARTONLY ]
```

<fileset-list>

is a list of disc files for which extents are to be allocated.

<num-extents>

is the total number of extents to be allocated to the file.

For nonpartitioned DP1 files, specify <num-extents> as a value in the range from 1 to 16, inclusive.

For nonpartitioned DP2 files, specify <num-extents> as a value in the range from 1 to <maximum-extents>. The value of <maximum-extents> is the number of extents set with the MAXEXTENTS file attribute when the file was created or last altered. The default value for <maximum-extents> is 16. See the description of the SET command in this section for more details.

For key-sequenced partitioned files, specify <num-extents> as a value in the range from 1 to 16; this is the number of extents to allocate in each partition.

For partitioned files that are not key sequenced, specify <num-extents> as a value in the range from 1 to 16 times the number of partitions. Sixteen extents are allocated for each partition, counting from the beginning of the file, until the total <num-extents> are allocated. (That is, extents 0-15 are in partition zero; extents 16-31 are in partition one; extents 32-47 are in partition two; and so on.)

→

PARTONLY

allocates extents to any primary and secondary partitions of partitioned files in <fileset-list>. If you omit PARTONLY, FUP allocates extents only to entire partitioned files, and only if their primary partition is in <fileset-list>. PARTONLY has no effect on nonpartitioned files.

Consideration

Note that <num-extents> can have a different significance for different types of files. For key-sequenced partitioned files only, it is the number of extents allocated in each partition. However, for nonpartitioned files and for partitioned files that are not key sequenced, <num-extents> is the total number of extents to be allocated to the file.

For example, to allocate an additional 12 extents to a nonpartitioned file that has 4 extents allocated, you should specify 16 for <num-extents>. To allocate an additional 12 extents for the primary partition of a key-sequenced partitioned file that has 4 extents allocated, specify 16 for <num-extents> and include the PARTONLY option so that the 12 extents are allocated only to the primary partition.

Example

Suppose that you want to create an unstructured DP2 file and allocate 10 file extents for the file. When the default file creation attributes are in effect, you create an unstructured file by entering:

```
-CREATE YRFILE
```

Initially, no file extents are allocated for the file, as shown by the FUP INFO, DETAIL command:

```
-INFO YRFILE, DETAIL
$BOOKS1.COMLANG.YRFILE 10/03/84 5:39
     TYPE U
     EXT ( 2 PAGES, 2 PAGES )
     MAXEXTENTS  16
     BUFFERSIZE  4096
     OWNER 8,44
     SECURITY (RWEP): NUNU
     MODIF: 10/01/84  10:38
     EOF 0  (0.0 % USED)
     EXTENTS ALLOCATED: 0
```

Note also that the extent size for YRFILE is not the default extent size for FUP, which is 1 page or 2048 bytes. At file creation, DP2 rounded the extent size upward to 2 pages or 4096 bytes. This change occurred because the extent size of DP2 files must always be an integral multiple of the BUFFERSIZE (for unstructured files) or of the BLOCK size (for structured files). To create an unstructured DP2 file with 1-page extents, you must specify a BUFFERSIZE of 2048 bytes with the FUP SET or FUP CREATE command.

You can preallocate 10 file extents for YRFILE by entering:

```
-ALLOCATE YRFILE, 10
```

Now, when you enter the FUP INFO YRFILE, DETAIL command, the following information is displayed:

```
$BOOKS1.COMLANG.YRFILE              10/03/84  5:41
     TYPE U
     EXT ( 2 PAGES, 2 PAGES )
     MAXEXTENTS  16
     BUFFERSIZE  4096
     OWNER 8,44
     SECURITY (RWEP): NUNU
     MODIF: 10/01/84  10:38
     EOF 0  (0.0 % USED)
     EXTENTS ALLOCATED: 10
```

ALLOW Command

Use the ALLOW command to set the number of errors and warnings that FUP allows when executing FUP commands. If this number is exceeded, the current FUP command is aborted.

If the error or warning limit is reached when you are entering FUP commands interactively, FUP displays a warning message, redisplays its hyphen prompt, and continues to accept commands. If, however, FUP is executing commands from a command file, FUP execution terminates when the error or warning limit is reached.

The syntax of the ALLOW command is:

```
ALLOW <num> { ERRORS   }
            { WARNINGS }
```

<num>

is the maximum number of ERRORS or WARNINGS that can occur before a FUP command is aborted. The default number of errors FUP allows is 0; the default number of warnings is 32,000.

Consideration

The ERRORS option does not apply to severe errors. For example, if you enter a FUP COPY command, and you get file-system error 11 (file or program not in directory) for the <in-file-name>, the COPY command terminates. Control of the terminal returns to the process through which you entered the COPY command.

Example

Suppose you are duplicating all the files in volume $BIG to volume $BACKUP. If you want FUP to warn you when 10 errors have occurred and to continue the DUP process, enter:

```
-ALLOW 10 ERRORS
-DUP $BIG.*.*, $BACKUP.*.*
```

ALTER Command

You can use the ALTER command to change selected characteristics of a disc file. ALTER affects the file's file label only; it does not create or purge files, and it does not insert, delete, or move records.

The syntax of the ALTER command is:

```
ALTER <file-name> { , <alter-option> }...

<file-name>

   is the name of the file whose characteristics are to be
   altered.  FUP expands a partial file name by adding the
   current default names for system, volume, and subvolume.

<alter-option>

   names the file characteristic to be altered.
   <alter-option> is any one of these:

      CODE <file-code>
      [ NO ] REFRESH
      ALTKEY ( <key-specifier> { , <altkey-param> }... )
      ALTFILE ( <key-file-number> , <file-name> )
      DELALTKEY <key-specifier>
      DELALTFILE <key-file-number>
      PART ( <sec-partition-num>
             , [ \<system-name>.] $<volume-name>
             [ , <pri-extent-size>
             [ , <sec-extent-size> ] ] )
      PARTONLY
      [ NO ] AUDIT
      MAXEXTENTS <maximum-extents>
      BUFFERSIZE <unstructured-buffer-size>
      [ NO ] BUFFERED
      [ NO ] AUDITCOMPRESS
      [ NO ] VERIFIEDWRITES
      [ NO ] SERIALWRITES
      RESETBROKEN
      ODDUNSTR
```

→

Parameters for All File Types

CODE <file-code>

alters the file code. Specify <file-code> as an integer in the range from 0 to 65,535, inclusive. File codes 100-999 are reserved for use by Tandem Computers Incorporated. Default <file-code> is 0.

[NO] REFRESH

causes the file's label to be copied to disc whenever the file's file control block (FCB) is marked as "dirty," such as when the end of file changes or when a free-space block change occurs. The file label is not written to disc if the only change is updating the file-label last modified field. Default is NO REFRESH (the label is not copied to disc).

Parameters for Files with Alternate-Key Fields

ALTKEY (<key-specifier> { , <altkey-param> }...)

adds, replace or alters an alternate-key specification.

<altkey-param> is any one of these:

FILE <key-file-number>
KEYOFF <key-offset>
KEYLEN <key-length>
NULL <null-value>
NO NULL
[NO] UNIQUE
[NO] UPDATE

For descriptions of <key-specifier> and <altkey-param>, see the SET command ALTKEY option in this section.

→

NOTE

If you add a new key specifier that references an undefined key-file number, you must include the ALTFILE option to define the alternate-key file.

ALTFILE (<key-file-number>, <file-name>)

adds or replaces the file name of an alternate-key file. You must include this parameter for any undefined key file number being referenced by an ALTKEY specification. Specify <key-file-number> as an integer in the range from 0 to 255, inclusive, designating the alternate-key file being named.

FUP expands a partial file name by adding the current default names for system, volume, and subvolume.

NOTE

If you add a new ALTFILE number, your ALTER command must also include an ALTKEY option specifying the alternate-key file.

If you are defining a new alternate-key file, you must create the file in a separate operation either before or after using ALTER ALTFILE.

DELALTKEY <key-specifier>

deletes an alternate-key specification. After you execute this option, you can no longer access the file through <key-specifier>, which is a two-byte value that you previously specified to uniquely identify this alternate-key field. (This is the value passed to the KEYPOSITION procedure when referencing this key field.)

→

Specify <key-specifier> as either:

"[c1]c2"

(a one- or two-character string inside quotation marks; if you give only one character, c1 is taken as a zero), or:

{ -32,768 : 32,767 }

(an integer in the range from -32,768 to 32,767, inclusive).

DELALTFILE <key-file-number>

deletes the reference to an alternate-key file. The alternate-key file must not be referenced by any existing <key-specifier>. The alternate-key file is not purged. After you execute an ALTER command with this option, the remaining key-file numbers and references to them are adjusted so that the set of key-file numbers for the file starts with 0, and the numbers are contiguous (that is, 0,1,...).

Parameters for Partitioned Files

```
PART ( <sec-partition-num>
       , [ \<system-name>.] $<volume-name>
       [ , <pri-extent-size>
       [ , <sec-extent-size> ] ] )
```

alters secondary partition specifications for partitioned files. You can specify extent sizes only if you are adding a new partition, or if you are altering an existing partition of a key-sequenced file. See the SET command PART option for a description of the PART parameters.

You can make the following alterations:

- For all file types, if the <sec-partition-num> already exists, you can change <volume-name>.

→

- For key-sequenced files only, you can redefine the extent sizes. (If the partition contains data, you must also perform other operations on the partition to accomplish the extent-size change.)

- For files that are not key sequenced, you can add a new, additional partition. (Partitions cannot be added to key-sequenced files.)

PARTONLY

means that any changes made with the ALTER command are made only to the file partition you specify.

[NO] AUDIT

designates the file as audited or nonaudited by TMF --see the Transaction Monitoring Facility (TMF) System Management and Operations Guide for details. If you specify AUDIT, FUP marks the file as an audited file. If you specify NO AUDIT, FUP marks the file as a nonaudited file. To use the AUDIT option, you must have purge access to a file (or be the super ID user).

Parameters for DP2 Files

MAXEXTENTS <maximum-extents>

- for nonpartitioned files only, sets the maximum number of extents to be allocated. Specify <maximum-extents> as an integer in the range from 1 to n, where n is a maximum value determined by the amount of free space in the file label. The absolute maximum is 978 bytes; the default is 16. FUP rounds upward to 16 any value you set between 1 and 15, inclusive. Before creating a file, you can set <maximum-extents> with the MAXEXTENTS attribute of the FUP SET command. At file creation, you can set <maximum-extents> with the MAXEXTENTS attribute of the FUP CREATE command.

→

BUFFERSIZE <unstructured-buffer-size>

for unstructured files only, is the internal buffer size to be used when accessing the specified file. You can set the BUFFERSIZE file attribute with the FUP SET, FUP ALTER, or FUP CREATE command. The actual buffer size used is rounded upward to the nearest valid DP2 block size. Valid block sizes are 512 bytes, 1024 bytes, 2048 bytes, and 4096 bytes. If you try to alter the BUFFERSIZE on a DP1 file, FUP gives the error message "applicable to DP2 files only."

[NO] BUFFERED

sets the mode of handling write requests to the file: using buffered or write-through cache. BUFFERED specifies a buffered cache; NO BUFFERED specifies a write-through cache. The default is BUFFERED for audited files or NO BUFFERED for nonaudited files.

CAUTION

If you use the buffered-cache option on a DP2 file that is not audited by TMF, a system failure or disc-process takeover can cause the loss of buffered updates to the file. Note also that an application program might not detect this loss or handle the loss correctly unless it is modified to do so.

[NO] AUDITCOMPRESS

for audited files, sets the mode of producing audit-checkpoint messages: using compressed messages or entire messages. Default is NO AUDITCOMPRESS.

[NO] VERIFIEDWRITES

sets the mode of file writes: verified or unverified. Default is NO VERIFIEDWRITES.

→

[NO] SERIALWRITES

sets the mode of writes to the mirror: serial or parallel. Default is NO SERIALWRITES.

RESETBROKEN

resets the file-label BROKEN flag for a nonaudited file. You should take steps to fix the broken parts of the file before using RESETBROKEN.

Parameter for Odd Unstructured Files

Unstructured ENSCRIBE files can be even or odd. With even unstructured files, any odd byte count you give for reading, writing, or positioning is rounded upward. (This is the default for unstructured files.) However, for odd unstructured files there is no upward rounding; you always read, write, or position at the byte count you give.

ODDUNSTR

changes an even unstructured file to odd unstructured. (To change an odd unstructured file to even, copy the odd file into a new file created as even unstructured.)

Considerations

- If you want to know the current file attributes for the file you want to alter, enter a FUP INFO command:

  ```
  -INFO <file-name> , DETAIL
  ```

- You must have both read and write access to alter a file.

- For DP2 files, changing the AUDIT option changes the default value of the BUFFERED attribute as well. If you specify NO AUDIT, the BUFFERED option and file label default is set to NO BUFFERED. If you specify AUDIT, the BUFFERED option is set ON. If, however, you have explicitly set the BUFFERED attribute, the value you set remains unchanged.

- If you include AUDIT, and the volume that contains the primary file or any of its secondary partitions or alternate-key files (containing at least one automatically updated alternate key) is not audited, the request fails with file-system error 80. But if these files are all audited, the labels of the alternate-key files are updated to reflect the audit option.

- To add an alternate key to a structured file that has no alternate-key file specified, your ALTER command must specify both the new alternate key and the new alternate-key file. (This is also true if you deleted an alternate key with the DELALTFILE option of the ALTER command.) For example, you can add the alternate key "aa" to the file FRED and specify the alternate-key file AFILE by entering:

```
-ALTER FRED, ALTFILE (0, AFILE), ALTKEY ("aa", FILE 0,&
-KEYOFF 0, KEYLEN 5)
```

Examples

The following ALTER command assigns file code 10 to MYFILE1:

```
-ALTER MYFILE1, CODE 10
```

This command causes the file label for MYFILE2 to be updated whenever the file's FCB changes:

```
-ALTER MYFILE2, REFRESH
```

This command assigns the alternate-key file MYFILE4 to MYFILE3 and gives it key-file number 2 (ALTFILE 2 must already be a defined attribute of MYFILE3):

```
-ALTER MYFILE3, ALTFILE (2 , MYFILE4)
```

The following command deletes the alternate key "ab" from MYFILE5:

```
-ALTER MYFILE5, DELALTKEY "ab"
```

BUILDKEYRECORDS Command

Use the BUILDKEYRECORDS command to generate the alternate-key records for specified key fields of a structured disc file and to write those records to a designated file.

The output of BUILDKEYRECORDS can be the actual destination alternate-key file; however, alternate-key loading is not as efficient as using a LOADALTFILE command. Usually, you use BUILDKEYRECORDS to generate the alternate-key records and store them in another file, then load them into the destination alternate-key file with a LOAD command. You can use this method (instead of doing a direct load of the alternate file with a LOADALTFILE command) when limited system resources do not permit a LOADALTFILE operation.

The syntax of the BUILDKEYRECORDS command is:

```
BUILDKEYRECORDS <primary-file-name> , <out-file-name>
                , <key-specifier-list> [ , <out-option> ]...
```

<primary-file-name>

names an existing primary file whose alternate-key records are to be generated. You must have previously defined alternate-key fields for the primary file.

<out-file-name>

names an existing file where the alternate-key records generated by this command are to be written. See the COPY command for characteristics of <out-file-name>.

→

<key-specifier-list>

names one or more alternate-key fields of the primary file whose corresponding alternate-key records are to be generated. For <key-specifier-list>, specify either of these two:

```
<key-specifier>
( <key-specifier> [ , <key-specifier> ]... )
```

<key-specifier>

is a two-byte value that uniquely identifies the alternate-key field. Specify either:

"[c1]c2"

(a one- or two-character string inside quotation marks; if you give only one character, then c1 is treated as a zero) or:

{ -32,768 : 32,767 }

(an integer in the range from -32,768 to 32,767, inclusive).

<out-option>

specifies the format and control of <out-file-name>. <out-option> is any one of these:

```
RECOUT <out-record-length>
BLOCKOUT <out-block-length>
PAD <pad-character>
EBCDICOUT
[ NO ] UNLOADOUT
[ NO ] REWINDOUT
SKIPOUT <num-eofs>
```

For an explanation of these options, see the descriptions of the COPY command <out-option> in this section.

Considerations

- BUILDKEYRECORDS causes the primary file to be read sequentially according to its primary-key field. For each record read from the primary file, BUILDKEYRECORDS generates one or more alternate-key records (corresponding to the number of key specifiers specified) and writes them to the out file. If you specify more than one key specifier, BUILDKEYRECORDS generates the corresponding alternate-key file records in the order of the key specifiers' ASCII collating sequence.
- BUILDKEYRECORDS honors any NULL specification defined for a key field; no alternate-key record is generated for a field that consists solely of a null character.
- BUILDKEYRECORDS ignores any NO UPDATE specifications.
- Any UNIQUE specifications are ignored; however, BUILDKEYRECORDS detects duplicate unique-key values when the alternate-key file is loaded.

Example

The following command generates the alternate-key records for MYFILE using key specifications "ab" and "cd" and writes the alternate-key records to $TAPE:

```
-BUILDKEYRECORDS MYFILE, $TAPE, ("ab","cd")
```

CHECKSUM Command

Use the CHECKSUM command to recompute the checksum value for blocks of data in disc files. You use this command when recovering from a software-detected checksum error. See your System Operator's Guide for more information.

The syntax of the CHECKSUM command is:

```
CHECKSUM <fileset-list> [ , PARTONLY ]

<fileset-list>

   is a list of files whose checksum values are to be re-
   computed.  CHECKSUM skips open files and warns you.

PARTONLY

   for partitioned files in <fileset-list>, recomputes
   checksum values for only the primary or secondary
   partitions and does not affect nonpartitioned files.
```

Considerations

- CHECKSUM reads each block of data specified by <fileset-list> and recomputes a checksum value for each file. However, CHECKSUM rewrites only those blocks which have changed since the last time the block's checksum value was computed.
- Checksum errors usually indicate a potential data integrity problem. CHECKSUM recomputes the checksum value for blocks of data but does not fix any data that might have changed.
- CHECKSUM aborts if a Peripheral Utility Program (PUP) DOWN or STOPOPENS command was executed for the volume named in <fileset-list>.

Example

To recompute checksum values for all files on volume $TEST1, enter:

```
-CHECKSUM $TEST1.*.*
```

COPY Command: Copy Form

Use the COPY command for either of these functions:

- To make a record-by-record copy between files (copy form of COPY)
- To display the contents of a file (display form of COPY)

The two forms of the COPY command are described in separate entries in this manual. This entry describes the copy form of COPY; the display form is described in the following entry. The command syntax given here covers both forms of the COPY command. The syntax given for the display form, however, defines the options for the display function only.

The syntax for both functions of the COPY command is:

```
COPY <in-file-name> [ , [ <out-file-name> ]
     [ , <copy-option> ]... ]
```

<in-file-name>

is the name of the file that is the source of the copy. This file can be a disc file, a nondisc device, or a process. (See "Considerations" for information regarding relative files.)

<out-file-name>

is the name of the file that is the destination of the copy. This file can be a nondisc device, a process, or an existing disc file. If you omit this option, the OUT file in effect for FUP is used (for example, if you are using FUP interactively, the OUT file is the home terminal).

→

<copy-option>

specifies the format of the input and output files. <copy-option> is any one of these:

```
FIRST {<ordinal-record-num>                          }
      { KEY { <record-spec> | <key-value> }          }
      { <key-specifier> ALTKEY <key-value>           }
COUNT <num-records>
UPSHIFT
UNSTRUCTURED
<in-option>
<out-option>
<display-option>
```

```
FIRST {<ordinal-record-num>                          }
      {KEY { <record-spec> | <key-value> }           }
      {<key-specifier> ALTKEY <key-value>            }
```

names the starting record of the input file for the copy. If you omit FIRST, the copy starts with the first record of the input file.

<ordinal-record-num>

is the number of records (from the beginning of the file) that are to be skipped. The first record in a file is record zero. If you specify this option for an unstructured disc file, the copy begins at:

<ordinal-record-num> * <in-record-length>

(Note that the actual reading of the <in-file-name> begins with the first record in the file.)

```
KEY {<record-spec>}
    {<key-value>  }
```

specifies the primary-key value for the starting record of a disc file. FUP performs a position operation to the file so that the actual read of the input file begins at the record you name with KEY. Specify <record-spec> as an integer in the range from 0 to 4,294,967,295, inclusive.

→

- For unstructured files, give the starting relative byte address for <record-spec>.
- For relative files, give the starting record number for <record-spec>.
- For entry-sequenced files, give <ordinal-record-num> for <record-spec> (see the FIRST option).

Use <key-value> to indicate the approximate position of the starting record for key-sequenced files. Specify <key-value> as either:

"c1c2...cn"

(a string of characters inside quotation marks) or:

```
{"c"          }  [, "c"          ]
{{ 0 : 255 }}  [, { 0 : 255 }]  ...
```

(a list of single characters, each inside quotation marks, and/or integers representing byte values in the range from 0 to 255, inclusive).

<key-specifier>

is a one- or two-character string, inside quotation marks, that designates the alternate key to be used for the positioning.

ALTKEY <key-value>

specifies the alternate key of the starting record for disc files. Because a position operation is done to the file, the actual read of the input file begins at the specified record. Specify <key-value> for key-sequenced files according to the description of <key-value> for the preceding parameter (KEY).

→

COUNT <num-records>

is the number of records to be copied. If you omit COUNT, FUP copies all records from the first record through the end of file (EOF).

UPSHIFT

makes lowercase alphabetic characters uppercase.

UNSTRUCTURED

for disc files only, identifies an unstructured disc file that is not in the format of the EDIT program nor of any ENSCRIBE file structure; FUP ignores the file structure.

For partitioned files, you must include this option if you want to copy partitions individually.

<in-option> is any one of these:

```
RECIN <in-record-length>
BLOCKIN <in-block-length>
TRIM <trim-character>
EBCDICIN
VARIN
SHARE
[ NO ] UNLOADIN
[ NO ] REWINDIN
SKIPIN <num-eofs>
REELS <num-reels>
```

(The <in-option> is not normally used with structured disc files.)

RECIN <in-record-length>

specifies the maximum number of bytes in an input record. The actual number of bytes in each input record (the read count) is dependent upon whether you also specify TRIM:

→

- If you do not specify TRIM, the read count is the actual number of bytes in the input record. (For unstructured files that are not in the format of the EDIT program, the read count is exactly <in-record-length> bytes, although the last read of the file might be less; for all other files, the read count is the number of bytes actually read.)

- If you specify TRIM, every trailing <trim-character> is deleted from the input record. The read count is a count of only the significant characters (those which have not been trimmed).

If you omit RECIN, FUP determines the <in-record-length> value as follows:

- If you specify an <in-block-length> less than or equal to 4096, that value is used for <in-record-length>. If you specify an <in-block-length> greater than 4096, the value of <in-record-length> is 4096.

- If you do not specify an <in-block-length>, and the input file is an unstructured file that is not in EDIT format or a process file, FUP uses 132.

- If you do not specify an <in-block-length>, and the input file is a structured disc file or a nondisc device, FUP uses the record length specified when the file was created (or at system generation).

BLOCKIN <in-block-length>

specifies the number of bytes in an input block. This number can be a value in the range from 1 to 32,767, inclusive. This is the actual number of bytes requested in a single physical read operation.

If <in-block-length> exceeds <in-record-length>, then input-record deblocking occurs. Records of the input-record length are extracted from the input block until <in-block-length> bytes are extracted or the last input record is encountered (the read count for all but possibly the last record in a block is equal to <in-record-length>). If <in-block-length> is not a multiple of <in-record-length>, then the last record extracted from a full block contains less than

→

<in-record-length> bytes (the read count is the number of bytes extracted).

If you omit the BLOCKIN specification, then FUP uses the <in-record-length> value for <in-block-length>; each input record is read in a separate physical operation; record deblocking does not occur.

TRIM <trim-character>

deletes any trailing characters matching <trim-character>. Specify <trim-character> as either:

"c"

(a single ASCII character in quotation marks) or:

{ 0 : 255 }

(an integer in the range from 0 to 255, inclusive, that specifies a byte value).

EBCDICIN

treats the file specified for <in-file-name> as if it contains EBCDIC characters and translates the characters to their ASCII equivalents.

In a conversion between EBCDIC and ASCII, all characters are converted into their own image, except these:

EBCDIC		ASCII
logical OR	<----->	exclamation point
cent sign	<----->	left square bracket
exclamation point	<----->	right square bracket
logical NOT sign	<----->	circumflex

VARIN

reads variable-length, blocked records. Such records can be produced using the COPY commands's VAROUT option, described under <out-option>.

→

Each variable-length, blocked record begins with a word that contains the length of the record; the read count equals the value of that length indicator. You cannot use the TRIM option with VARIN.

SHARE

for disc files only, opens <in-file-name> with a shared exclusion mode. Using SHARE, you can copy a file even if it is currently open by another process (unless it is open with exclusive exclusion mode). If you omit SHARE and if <in-file-name> is a disc file, the file is opened with protected exclusion mode.

[NO] UNLOADIN

for magnetic tape only, specifies that the tape is unloaded (or not unloaded) when rewinding occurs. The default is UNLOADIN (the tape is unloaded when rewound).

[NO] REWINDIN

for magnetic tape only, specifies that the tape is rewound (or not rewound) when the end of file is read from tape. If you specify NO REWINDIN, the tape remains positioned. The default is REWINDIN (the tape is rewound).

SKIPIN <num-eofs>

for magnetic tape only, means that the tape is moved past <num-eofs> end-of-file (EOF) marks before the data transfer begins. Specify <num-eofs> as an integer in the range from -255 to 255, inclusive.

- If you specify a positive value for <num-eofs>, the tape is wound forward past <num-eofs> EOF marks, and is positioned immediately after the last EOF mark passed.
- If you specify a negative value for <num-eofs>, the tape is wound backwards over (-1 times <num-eofs>)

→

EOF marks, then moved forward and positioned immediately ahead of the last EOF mark passed.

- If you specify a value of 0 for <num-eofs>, the SKIPIN option is ignored.
- If you omit the SKIPIN option, the tape remains at its current position, and the data transfer begins with the next physical record on tape.

REELS <num-reels>

for magnetic tape only, sets the number of reels made up by <in-file-name>. Specify <num-reels> as an integer in the range from 1 to 255, inclusive. Tape is read until <num-reel> occurrences of two consecutive EOF marks (denoting end of reel) are reached. At each end of reel before the last reel, the tape is rewound and unloaded, and you are prompted for the next reel.

If you omit REELS, <in-file-name> data transfer terminates when a single EOF mark is encountered.

<out-option> is any one of these:

```
RECOUT <out-record-length>
BLOCKOUT <out-block-length>
PAD <pad-character>
EBCDICOUT
VAROUT
FOLD
[ NO ] UNLOADOUT
[ NO ] REWINDOUT
SKIPOUT <num-eofs>
```

(The <out-option> is not normally used with structured disc files.)

RECOUT <out-record-length>

sets the maximum number of bytes in an output record. The actual number of bytes written for each output record (the write count) is dependent upon whether you also specify PAD:

→

- If you do not specify PAD, then the write count is either the read count or <out-record-length>, whichever is less.
- If you specify PAD, then the write count is <out-record-length>; if the number of input bytes is less than <out-record-length>, then the record is padded with trailing pad characters.
- In either case, if the number of input bytes exceeds <out-record-length>, the input record is truncated at output-record length bytes (unless you specify FOLD).

If you omit RECOUT, FUP determines the <out-record-length> value as follows:

- If you specify an <out-block-length> less than or equal to 4096, the value of <out-block-length> is used for <out-record-length>. If <out-block-length> is greater than 4096, <out-record-length> is 4096.
- If you do not specify <out-block-length>, and if <out-file-name> is an unstructured disc file or a process file, <out-record-length> is 132.
- If you do not specify <out-block-length>, and if <out-file-name> is a structured disc file or a nondisc device, <out-record-length> is the record length specified for the file at file creation (or at system generation).

BLOCKOUT <out-block-length>

sets the number of bytes in an output block. Specify <out-block-length> as an integer in the range from 1 to 32,767. This is the maximum number of bytes to be written in a single physical operation.

If the output-block length is greater than the output-record length, then output-record blocking occurs. The block is filled with <out-record-length> records until it contains <out-block-length> bytes or the last output record encountered. If <out-block-length> is not a multiple of <out-record-length>, then the last record in a full block is truncated. If the write count for a

→

record is less than <out-record-length>, the output record is padded in the output block with trailing nulls. The actual number of bytes written in a physical operation is <out-block-length> for all blocks but the last block. If the last block is not full, then the actual number of bytes written is equal to the number of records in the last block times the <out-record-length>.

If you omit BLOCKOUT, FUP uses the write-count value for <out-block-length>; each output record is written in a separate physical operation; record blocking does not occur.

PAD <pad-character>

means that records containing less than <out-record-length> bytes are padded with the <pad-character> up to the specified record length. Specify <pad-character> as:

"c"

(a single ASCII character inside quotation marks) or:

{ 0 : 255 }

(an integer in the range from 0 to 255, inclusive, specifying a byte value).

If you omit <pad-character>, FUP uses an ASCII null character.

EBCDICOUT

means that output characters are translated to their EBCDIC equivalents. If you omit EBCDICOUT, FUP does not translate output.

→

In a conversion between ASCII and EBCDIC, all characters are converted into their own image, except for the following:

ASCII		EBCDIC
exclamation point	<----->	logical OR
left square bracket	<----->	cent sign
right square bracket	<----->	exclamation point
circumflex	<----->	logical NOT sign

VAROUT

writes variable-length, blocked records.

Each variable-length record is preceded by a one-word record-length indicator. The record-length word contains the record's length in bytes. The record-length word is always aligned on a word boundary, although records might contain an odd number of bytes.

A record's record-length word and the write count are equal. However, it is possible that the record has been truncated. Truncation occurs if the record is longer than RECOUT or longer than BLOCKOUT minus 2 (two extra bytes are required for the record-length word). Blocks cannot be spanned. If the next record with its record-length word does not fit into the current block, VAROUT terminates the current block and begins a new block.

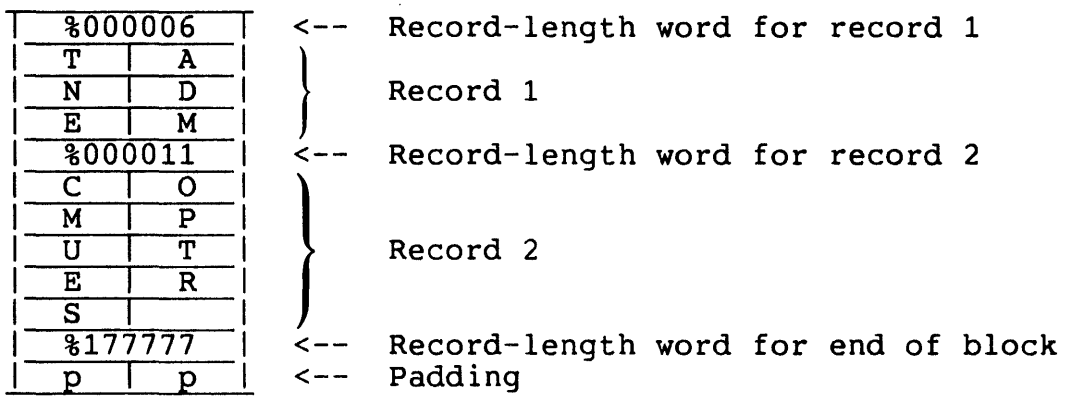
VAROUT terminates a block by writing a record-length word of -1 (%177777) to indicate that there are no more valid records in the block and then padding out the remainder of the physical block. VAROUT cannot write the record-length word of -1 when the previous record ends on the block boundary, or when <out-block-length> is odd and only one byte remains in the block.

Null records (records with a length of zero) are supported.

The PAD and FOLD options are not allowed with VAROUT.

The following sample block has a BLOCKOUT length of %30. The three records "TANDEM", "COMPUTERS", and "INC." illustrate the action of VAROUT:

→



Record 3 (INC.) and its record-length word require six bytes beginning on a word boundary. Since only four bytes remain in the sample block, VAROUT terminates the block; record 3 is written to the next block.

FOLD

divides input records that are longer than <out-record-length> into as many <out-record-length> records as needed to copy the entire input record. If the last record written because of a FOLD is shorter than <out-record-length>, and if you specify PAD, padding occurs. If you omit FOLD, truncation might occur.

[NO] UNLOADOUT

for magnetic tape only, means that the tape is unloaded (or not unloaded) when rewinding occurs. The default is UNLOADOUT (the tape is unloaded when it is rewound).

[NO] REWINDOUT

for magnetic tape only, means that the tape is rewound when the COPY command completes. If you specify NO REWINDOUT, then the tape remains positioned. The default is REWINDOUT (the tape is rewound).

→

SKIPOUT <num-eofs>

for magnetic tape only, means that the tape is moved past <num-eofs> end-of-file (EOF) marks before the data transfer begins. Specify <num-eofs> as an integer in the range from -255 to 255, inclusive.

- If you specify a positive value for <num-eofs>, the tape is wound forward past <num-eofs> EOF marks, and is positioned immediately after the last EOF mark passed.
- If you specify a negative value for <num-eofs>, the tape is wound backwards over (-1 times <num-eof>) EOF marks, then moved forward so that it is positioned immediately ahead of the last EOF mark passed.
- If you specify a value of 0 for <num-eofs>, the SKIPOUT option is ignored.
- If you omit the SKIPOUT option, the tape remains at its current position, and the data transfer begins with the next physical record on tape.

NOTE

The display form of the COPY command and the <display-option> are detailed in the next syntax entry in this section.

Considerations

- If the destination of the copy is an unstructured file, a relative file, or an entry-sequenced file, data is appended to the end of file. Existing data is not overwritten.
- If the <in-file-name> is in the format of the EDIT program, it is treated like a structured file; each text line is treated as a logical record and therefore has a count read attribute. (This differs from unstructured files, where each physical read, except possibly the last read of a file, returns exactly <in-record-length> bytes.)

- The COPY command cannot be used to write output files in the format of the EDIT program.

- You cannot use the COPY command to copy a multireel set of tapes created by a Tandem COBOL program, because Tandem COBOL marks the end of a multireel set of tapes differently than FUP does. You must use a COBOL program to copy such tapes. See the COBOL Reference Manual for more information.

- Although you can specify a block size up to 32,767 bytes for the BLOCKIN and BLOCKOUT parameters, some peripheral devices have smaller maximum block sizes that must not be exceeded when using the COPY command.

- If you copy a file that contains data records and zero-length records (empty records) to a relative output file, all records are written (including the zero-length records), unless the input file is also a relative file. If the input file is a relative file, the zero-length records are skipped. For example, when you copy a relative file containing a combination of eight data records and two zero-length records, the record count of the output file is eight records instead of ten. Thus, if you copy data with zero-length records in and out of a relative file, you lose the zero-length records.

- FUP COPY uses sequential block buffering when accessing <in-file-name>. As a result, all record locks are ignored. (See the file-access section in the ENSCRIBE Programming Manual.)

- When you enter a COPY command with no options, FUP opens <in-file-name> with read-only access mode, protected exclusion mode (unless <in-file-name> is a terminal, which is always opened with shared exclusion mode). If, however, you include the SHARE option in the command, and <in-file-name> is a disc file, <in-file-name> is opened with shared exclusion mode. FUP opens <out-file-name> with exclusive exclusion mode (unless <out-filename is a terminal).

- A physical read of <in-file-name> reads one logical record. (A logical record is 132 bytes for unstructured files or files in the format of the EDIT program; for structured disc files and nondisc devices, a logical record is the record length you specify.) A physical write to the <out-file-name> writes one logical record (the actual number of bytes written is the number of bytes read).

- If the AUDIT option is set for the <out-file-name>, the copy request fails with file-system error 75.

- Be careful when using the PAD and TRIM options in a FUP COPY or LOAD operation. If your data contains the <trim-character> or <pad-character>, data might be altered or lost. Suppose you pad each record in a data file with zeros to a standard size in bytes, and then store the records in another file. If you later trim the trailing zeros when you FUP COPY or LOAD the stored records, any original data that ended with zero is trimmed. To avoid this problem, use a <pad-character> or <trim-character> that is not contained in your data.

Example

This example copies the first 500 records of MYFILE to YOURFILE (two unstructured files in the current default subvolume). Input records are 80 bytes, and the copy is done in 800-byte physical reads:

```
-COPY MYFILE, YOURFILE, COUNT 500, RECIN 80, BLOCKIN 800
```

COPY Command: Display Form

Use the display form of the COPY command to display the contents of a file. This syntax description explains the display options only. For the complete syntax, see "COPY Command: Copy Form."

The syntax of the COPY command display options is:

```
COPY <in-file-name> [ , [ <out-file-name> ]
     [ , <display-option> ]... ]
```

<display-option>

specifies the format for displaying the file's contents. If you omit <display-option>, no formatting or conversion occurs; each record is displayed exactly as read from the <in-file-name>.

The <display-option>s are:

```
O[CTAL]
D[ECIMAL]
H[EX]
BYTE
A[SCII]
NO HEAD
```

OCTAL or O

displays the contents of the file in octal format and in ASCII format.

DECIMAL or D

displays the contents of the file in decimal format and in ASCII format.

→

HEX or H

displays the contents of the file in hexadecimal format and in ASCII format.

BYTE

displays the contents of the file in byte format and ASCII format. The two bytes of each word are converted separately.

- If you do not specify BYTE, a word is treated as a single value and then converted accordingly.
- If you specify BYTE but neither OCTAL nor DECIMAL nor HEX, the display is in byte-octal format.

ASCII or A

displays the contents of the file in ASCII format. This option has no meaning when combined with any preceding <display-option>.

NO HEAD

omits the heading preceding each record displayed.

Listing Format

Figure 3-1 shows the format used by the FUP COPY command for displaying files.

```
[ <file-name> RECORD <rec-num> KEY <xx> ( %<yy> )
                  LEN <length-value> <mm/dd/yy> <hh:mm> ]

 <offset>  <word 0> <word 1> ... <word n>  [ <ASCII format> ]
     .         .       .             .     [         .       ]
     .         .       .             .     [         .       ]
     .         .       .             .     [         .       ]
```

Figure 3-1. FUP COPY Command Listing Format

The headers and variables in Figure 3-1 are:

<file-name>	is the name of the file being displayed.
RECORD <rec-num>	indicates the ordinal number of the record that follows.
KEY <xx> (%<yy>)	for disc files, indicates the record's primary-key value (current-record pointer) in decimal (<xx>) and octal (%<yy>). For unstructured files, <xx> is a relative byte address; for relative files, <xx> is a record number; for entry-sequenced files, <xx> is a record address; for key-sequenced files, <xx> is not given.
LEN <length-value>	indicates the decimal length of the record in bytes.
<mm/dd/yy>	is the current date (displayed for the first record only).
<hh:mm>	is the current time (displayed for the first record only).

<offset>	is the offset from the beginning of the record of word 0 in the line to the right. If you specify BYTE, then offset is a byte offset; otherwise, a word offset is given. <offset> is given in the same number base as the associated line.
<word 0>...<word n>	is a block of n contiguous words of <file-name>. If <out-record-length> is less than 100, n equals 7; if <out-record-length> is between 100 and 120, n equals 9; otherwise, n equals 11.
<ASCII-format>	is the ASCII representation of the line to the left. Nonprintable characters are represented by a period (.). If only ASCII or A is specified, the display is in ASCII format. The offset is given in decimal representation.

Consideration

If you specify more than one of OCTAL, DECIMAL, and HEX, each line is displayed in each specified format. The lines are in ascending order of base values. That is, O equals 8, D equals 10, H equals 16, and so forth.

Example

This example displays the contents of MYFILE, a file in the default volume and subvolume. The display is in both octal and ASCII format:

```
-COPY MYFILE,, OCTAL
```

CREATE Command

Use the CREATE command to create a disc file having the current file-creation attributes (see the SET command). To create a file with other attributes, you must include <create-param>s in your CREATE command to set the attributes you want.

The syntax of the CREATE command is:

```
CREATE <file-name> [ , <create-param> ]...
```

<file-name>

is the name of the file to be created. A partial file name is expanded using the current default names for system, volume, and subvolume.

<create-param>

if included, overrides the corresponding current file-creation attribute setting for this creation. See the SET command for the <create-param>.

Considerations

- For a structured file, extents must be at least as large as the block size--otherwise multiple extents are allocated each time a block is required.
- You must create primary files and alternate-key files separately to take advantage of the REFRESH, DCOMPRESS, and ICOMPRESS options of the CREATE command. The CREATE command passes these attributes only to the primary file when it is created.

- If you enter a CREATE, AUDIT command, and the volume where the primary partition, secondary partitions, or alternate-key files reside is not audited, then file-system error 80 is returned. However, if you create an alternate-key file with the NO UPDATE option, and this key file is not on an audited volume, the error does not occur.

- If FUP terminates because an error occurs while CREATE is automatically creating alternate-key files, the primary-key file and any alternate-key files already created are not deleted. You must manually purge the files.

- In DP2 key-sequenced files, index and data blocks are the same size. For this reason, if you include an IBLOCK specification for a DP2 key-sequenced file, CREATE ignores the index block size you give.

Examples

This example creates a file named MYFILE in the current default volume and subvolume. The file has the current file-creation attributes (as shown by the SHOW command):

```
-CREATE MYFILE
```

If you include the <create-param> in your CREATE command, the values you specify override the current file-creation attributes. For example, regardless of the current attributes, this command creates a relative file whose record length is 10, with an alternate key defined on the first five bytes of the record, and with the alternate key residing in the alternate-key file SECFILE:

```
-CREATE MYFILE, TYPE R, REC 10, ALTKEY ("AA", FILE 0, &
-KEYLEN 5) , ALTFILE (0, SECFILE)
```

DEALLOCATE Command

Use the DEALLOCATE command to deallocate any file extents beyond the one that includes the end-of-file (EOF) address of the specified disc files.

The syntax of the DEALLOCATE command is:

```
DEALLOCATE <fileset-list> [ , PARTONLY ]
```

<fileset-list>

is a list of files whose unused extents, past the EOF extent, are to be deallocated.

PARTONLY

means that any unused extents for any primary or secondary partitions of partitioned files that reside in <fileset-list> are to be deallocated. If you omit PARTONLY, extents are deallocated for entire partitioned files only, and only if the primary extent resides in <fileset-list>. PARTONLY has no effect on nonpartitioned files.

Example

This example deallocates the unused extents past the EOF of MYFILE:

```
-DEALLOCATE MYFILE
```

DUP[LICATE] Command

Use the DUP (or DUPLICATE) command to make a copy of a disc file. In a single FUP DUP command, you can copy several sets of files.

The syntax of the DUP command is:

```
DUP[LICATE] <from-fileset-list> , <to-fileset>
            [ , <rename-option> ] ...
            [ , <file-conversion-option> ] ...
            [ , NEW    ]  [ , PARTONLY ]  [ , SAVEID     ]
            [ , OLD    ]                  [ , SOURCEDATE ]
            [ , PURGE  ]                  [ , SAVEALL    ]
            [ , KEEP   ]
```

<from-fileset-list>

is a <fileset-list> specifying the files to be duplicated. FUP duplicates files one at a time. Partial file names are expanded using the current default system, volume, and subvolume where necessary.

<to-fileset>

is a <fileset> specifying the destination of the duplication.

- To duplicate more than one file, you must specify the disc file name part of <to-fileset> as "*"; each output file is given the disc file name of its corresponding input file.
- If you specify the <subvolume-name> of <to-fileset> as "*", then each output file is given the <subvolume-name> of its corresponding input file.

The AUDIT options of the files in <to-fileset> are reset regardless of the state of that option for the files in <from-fileset-list>.

→

<rename-option>

renames the secondary-partition volume and alternate-key files at the time of the duplication; FUP creates the destination file with new names for the secondary-partition volumes and alternate-key files. This option applies only when FUP creates the destination file (that is, when either the NEW or the PURGE option is in effect).

<rename-option> can be either of these:

```
PART ( <sec-partition-number>
     , [ \<system-name>.] $<volume-name>
     [ , <pri-extent-size> [ , <sec-extent-size> ] ] )
ALTFILE ( <key-file-number> , <file-name> )
```

```
PART ( <sec-partition-number>
     , [ \<system-name>.] $<volume-name>
     [ , <pri-extent-size> [ , <sec-extent-size> ] ] )
```

specifies a new name for a secondary partition of a partitioned file. See the description of the PART option in the FUP SET command for a description of <sec-partition-number>, <pri-extent-size>, and <sec-extent-size>.

If you specify PART, the name of the partition (indicated by <sec-partition-number>) is replaced by the new partition name you specify. The indicated <sec-partition-number> should already exist in the source file; if it doesn't, FUP issues a warning.

If FUP is performing a file conversion from DP1 to DP2 or vice versa, you can also use the PART option to specify the destination partition's extent sizes. Then the extent sizes you specify will override FUP's calculations (see the EXT option in this syntax description and the "Considerations" that follow). FUP ignores extent-size specifications if no file conversion is taking place.

You cannot use PART to specify a new key separator for key-sequenced partitioned files.

→

ALTFILE (<key-file-number> , <file-name>)

specifies a new name for an alternate-key file. The ALTFILE parameters are described in the SET command description for ALTFILE options. The name of the alternate-key file (indicated by <key-file-number>) is replaced with the new <file-name>. The indicated <key-file-number> should already exist in the source file; if it doesn't, FUP issues a warning.

<file-conversion-option>

overrides standard conversion settings when converting files from DP1 format to DP2 format, or vice versa.

<file-conversion-option> is any one of these four:

```
EXT {<extent-size>                                  }
    {( <pri-extent-size> , <sec-extent-size> )      }
SLACK <percentage>
DSLACK <percentage>
ISLACK <percentage>
```

```
EXT {<extent-size>                                  }
    {( <pri-extent-size> , <sec-extent-size> )      }
```

sets the extent size. When you DUP a file from a DP1 volume to a DP2 volume or vice versa, FUP might need to recalculate extent sizes because of the differing data capacities and extent-size restrictions for each disc process type. Use the EXT parameters to override FUP's calculation of extent sizes. FUP ignores EXT specifications if no file conversion takes place.

See the description of the EXT option in the FUP SET command for a description of <extent-size>, <pri-extent-size>, and <sec-extent-size>.

For partitioned files, use the PART option to specify the extent sizes of secondary partitions.

→

SLACK <percentage>
DSLACK <percentage>
ISLACK <percentage>

specify slack values for duplicating key-sequenced files from a DP1 volume to a DP2 volume, or vice versa. FUP uses the specified slack values when loading the destination file with data from the source file. See the FUP LOAD command for a description of these options.

NEW

means that each file specified by <to-fileset> does not exist yet. For each file in <from-fileset-list>, FUP creates a new file with characteristics identical to the corresponding input file. (NEW is the default mode.)

OLD

means that each file specified by <to-fileset> already exists. The characteristics of each file in <to-fileset> must match the characteristics of the corresponding file in <from-fileset-list> in the following areas (if applicable): file type, record size, data- and index-block sizes, data and index compression mode, and key length and key offset. If an object file is in <from-fileset-list>, then the extent size must match also. What you specify for <to-fileset> must be large enough to contain the <from-fileset> data. Existing data in <to-fileset> is overwritten.

KEEP

does not duplicate any files that exist in both the <to-fileset> and <from-fileset-list>. If a file specified in <to-fileset> already exists, then the corresponding file in <from-fileset-list> is not duplicated. Files that do not already exist in <to-fileset> are duplicated. (FUP lists the names of the files not duplicated.)

→

PURGE

purges any duplicate destination files before making the duplication. (If a file currently exists with the same name as a destination of the duplication, FUP purges the current contents of the file.) FUP creates a new file, with identical characteristics, for each file in <from-fileset-list>.

NOTE

NEW is implied if OLD, PURGE, and KEEP are omitted.

PARTONLY

specifies that if <from-fileset-list> defines any primary or secondary partitions of a partitioned file, only those partitions and any nonpartitioned files are to be duplicated; entire partitioned files are not duplicated.

Secondary partitions defined by <from-fileset-list> are not duplicated unless you specify PARTONLY.

If you omit PARTONLY and define a primary partition in <from-fileset-list>, FUP duplicates the entire partitioned file (that is, all partitions).

If the DUP operation converts files from a DP1 volume to a DP2 volume or vice versa, you cannot use PARTONLY if data is redistributed among partitions. This can happen with partitioned relative and entry-sequenced files if the data capacities of the source and destination are not the same.

SAVEID

transfers the owner ID and security of the files named in <from-fileset-list> to <to-fileset> with no changes. Without SAVEID, <to-fileset> gets the owner and default security of the user executing the DUP command.

→

SOURCEDATE

transfers the timestamp of the files in <from-fileset-list> to <to-fileset> unchanged.

SAVEALL

transfers the owner ID, security, and timestamp of the files in <from-fileset-list> to <to-fileset> unchanged. Without SAVEALL, <to-fileset> gets the owner and default security of the user executing the DUP command, and the timestamp is the time the command was issued.

Considerations

- The DUP command opens the file to be duplicated with read-only access and with protected exclusion mode. See the ENSCRIBE Programming Manual for information about file access and exclusion modes.
- If you specify OLD and name a secondary partition but do not specify PARTONLY, an error occurs if you attempt to duplicate a partitioned file.
- If you DUP a file that is audited by the Transaction Monitoring Facility, the new file is not an audited file. Use an ALTER AUDIT command to restore the file's audited status.
- If you DUP a file with local security (such as "AOAO") to a remote system, the new file's security is set to allow comparable network access (such as "NUNU").
- The SOURCEDATE and SAVEALL options preserve the timestamp of the files in <from-fileset-list> only if the target file resides in the same system where FUP is running.
- The SAVEID and SAVEALL options:

  --Preserve the licensed attribute (see the LICENSE command) only if FUP's process accessor ID is the super ID, and the target file resides on the system where FUP is running (no warning message is given if the target file is remote)

--Preserve the PROGID attribute (see the SECURE command) if FUP's process accessor ID is the super ID or if you are the owner of the source file, and the target file resides on the system where FUP is running (no warning message is given if the target file is remote)

--Transfer the CLEARONPURGE attribute (see the SECURE command) to a local or remote target file (the target file need not be on the system where FUP is running)

Considerations for DP1/DP2 File Conversion with FUP DUP

FUP performs file conversion when duplicating files from one disc-process format to the other (that is, when duplicating files from a DP1 volume to a DP2 volume, or vice versa).

- In file conversion, FUP normally recalculates the extent sizes so that the data capacities of the source and destination files are approximately equal despite differences between the disc-process types. You can use the EXT or PART options to override the calculation of extent sizes done by FUP. You cannot, however, override the calculation by FUP if you are duplicating an unstructured partitioned file, or if you intend to convert PARTONLY any relative or entry-sequenced files.

- You cannot use the PARTONLY option if, on conversion, data is redistributed among partitions. This can happen with partitioned relative and entry-sequenced files if the data capacities of the source and destination partitions are not exactly the same. Note this general rule: if the primary partition can be converted PARTONLY, all the secondary partitions can also be converted PARTONLY.

- If you specify extent sizes (with the EXT or PART options) when converting a file PARTONLY, you must ensure that the actual extent sizes of the partitions are the sizes indicated in the file label of the primary partition.

- In a file conversion, block sizes change for structured DP1 files with block sizes that are not shared by DP2. Blocks of 1536 bytes are changed to 2048 bytes. Blocks of 2560, 3072, and 3584 bytes are changed to 4096 bytes. Each block-size change is flagged with a warning message.

- For DP1 files, data blocks and index blocks can be different sizes. For DP2 files, however, the index-block size is equal to the data-block size. In a DUP conversion from DP1 to DP2,

FUP uses as the block size the larger of the data-block and index-block sizes, rounded upward to a valid DP2 block size (that is, 512, 1024, 2048, or 4096 bytes).

- FUP attempts to convert relative and entry-sequenced files so that the record addresses of records in the source and destination remain the same. If, however, the block size for an entry-sequenced file changes, then the record addresses of source and destination will differ. FUP issues a warning; you should reload the alternate-key files of entry-sequenced files whose block sizes change.

- When FUP converts a partitioned unstructured file from DP1 to DP2, the destination is always created with a BUFFERSIZE of 2048 bytes. The BUFFERSIZE of nonpartitioned files, however, can be either 2048 bytes or 4096 bytes because FUP attempts to keep the same extent sizes in a file conversion.

Examples

This example duplicates all the files in subvolume MYSVOL as well as all the files in the current default subvolume, and places the copies, with the same file names, in the NEWSVOL subvolume:

```
-DUP (MYSVOL.*, *), NEWSVOL.*
```

The following command performs a DP1/DP2 file conversion by duplicating the key-sequenced file $DP1.RECORD.DATA to $DP2.RECORD.DATA:

```
-DUP $DP1.RECORD.DATA, $DP2.*.* , ALTFILE (0, $DP2.RECORD.&
-ALTFILE), EXT (2, 3), DSLACK 20, ISLACK 10
```

The preceding command also does the following:

- Renames the first alternate-key file (file 0) of the destination file $DP2.RECORD.ALTFILE

- Specifies primary and secondary extent sizes of the DP2 file at 2 pages and 3 pages, respectively; because this command explicitly sets the extent sizes, no upward rounding occurs; otherwise, FUP would change the extent size to fit all source data into the destination file

- Sets the data and index slack with which records are loaded into the destination file at 20 percent and 10 percent, respectively

EXIT Command

Use the EXIT command to stop the current FUP process and return to the command interpreter.

The syntax of the EXIT command is:

```
EXIT
```

Considerations

- The FUP process terminates when it reads the end of file (EOF) of the IN <file-name> you specified in your command to run FUP. For this reason, you do not have to end a FUP command file with an EXIT command.
- Entering CTRL/Y at the terminal is the same as an end of file. If you type CTRL/Y at FUP's prompt, FUP displays "EOF!" and terminates.

Example

This example illustrates terminating FUP. Control of the terminal returns to the command interpreter:

```
-EXIT
:
```

FC Command

Use FC (the Fix Command) to modify and resubmit the last command line you entered. FC uses the R, I, and D subcommands as described for the FC command in COMINT (see Section 2 for details).

The syntax of the FC command is:

```
FC
```

Example

In this example, the DETAIL option in the first command is changed to the STATISTICS option:

```
-FC
-INFO MYFILE, DETAIL
.             ddSTAT
-INFO MYFILE, STAT
.
```

FILES Command

Use the FILES command to display the file names associated with one or more subvolumes.

The syntax of the FILES command is:

```
FILES [ / OUT <list-file> / ] [ <subvolset> ]
```

OUT <list-file>

names an existing file or a device that will receive the output of the FILES command.

<subvolset>

names a subvolume or set of subvolumes whose file names are to be listed. If you omit <subvolset>, FUP lists the files that reside in the current default volume and subvolume. <subvolset> has the form:

```
[ \ <system-name>.][ $<volume-name>.]{<subvolume-name>}
                                     {      *       }
```

- If you omit \<system-name>, FUP lists the designated file names on the current default system.
- If you omit $<volume-name>, FUP lists the designated file names on the current default volume.
- If you use "*" in place of <subvolume-name>, FUP lists all the file names on the designated volume.
- If you omit both <subvolume-name> and "*", FUP lists the file names in the current default subvolume on the designated volume.
- If you include both the volume and the subvolume, <subvolset> must be a contiguous string with <volume-name> separated from the other part by a period (.).

→

- If you include only $<volume-name>, omit the period (.) at the end.

Examples

To display the names of the files in your current default subvolume, enter:

```
-FILES
```

This command lists the names of the files in each subvolume of the current default volume:

```
-FILES *
```

To list the files in the current default subvolume on volume $DATA, enter:

```
-FILES $DATA
```

This command lists the names of all the files in each subvolume on the volume $VOL2:

```
-FILES $VOL2.*
```

The next example displays the names of all files in the JIMMY subvolume of the $ACES volume on the \TENNIS system:

```
-FILES \TENNIS.$ACES.JIMMY
```

GIVE Command

Use the GIVE command to change the owner of a file. Only the file's current owner (or the super ID user) can execute the GIVE command for any given file.

The syntax of the GIVE command is:

```
GIVE <fileset-list> , <group-id> , <user-id> [ , PARTONLY ]

<fileset-list>

   is a list of files whose ownership is to be given away.

<group-id> , <user-id>

   is the group number and user number of the user who is
   to be given ownership of the file.

PARTONLY

   for partitioned files, specifies that only partitions
   included in the <fileset-list> are to be given.
   If you omit PARTONLY, only entire partitioned files whose
   primary partitions reside in <fileset-list> are given.
   PARTONLY has no effect on nonpartitioned files.
```

Considerations

- You cannot use the GIVE command on a file that is currently open with exclusive exclusion mode.
- If a process has a file open when that file is given to another user, the access rights of the process are not affected.

- If you give a program file whose PROGID bit is set, the GIVE command clears that bit. (You can set the PROGID bit with the SETMODE function or with the FUP SECURE command; if this bit is set, the accessor's ID is set to the program file's ID when the program is run. For more information, see the GUARDIAN Operating System Programmer's Guide.)

- Files that you give to another user remain in their original subvolume. To move a copy of a file to another subvolume, use FUP DUP.

Examples

This command gives ownership of all files in the current default subvolume to the user with user ID 8,1:

```
-GIVE *, 8,1
```

To give the files PROG1, PROG2, and LIB in the subvolume $WORK.ORG to the user whose user ID is 4,10, enter:

```
-GIVE ($WORK.ORG.PROG1, $WORK.ORG.PROG2, $WORK.ORG.LIB), 4,10
```

HELP Command

Use the HELP command to list the syntax of FUP commands.

The syntax of the HELP command is:

```
HELP [ / OUT <list-file> / ]  [<command-name>]
                              [ALL            ]
```

OUT <list-file>

names a file to receive the output of the HELP command. If you omit this option, the output is sent to the OUT <list-file> in effect for the current FUP (usually this is the home terminal).

<command-name>

is the name of a FUP command whose syntax you want to see.

ALL

lists the names of all FUP commands.

Examples

Enter HELP to display the syntax of the FUP HELP command:

```
-HELP
HELP [ / OUT <list file> / ] [ <command name> ]
                             [ ALL            ]
```

This HELP command displays the syntax of the FUP CREATE command:

```
-HELP CREATE
CREATE <file name> [ , <create param> ] ...
   <create param>, see "SET"
```

INFO Command

Use the INFO command to display disc file characteristics.

The syntax of the INFO command is:

```
INFO [ / OUT <list-file> / ] <fileset-list>
     [, DETAIL                                    ]
     [, STAT[ISTICS] [ , PARTONLY ]               ]
     [, EXTENTS                                   ]
     [, USER  {<group-id> , <user-id>            } ]
     [        {<groupname>.<username>            } ]
```

OUT <list-file>

names an existing disc file or a device to receive the listing output of the FUP INFO command. By default, the OUT <list-file> is the home terminal. (See Figure 3-2 for the FUP INFO listing format.)

DETAIL

gives detailed information on file characteristics. (See Figure 3-3 for the DETAIL option listing format.)

STAT[ISTICS]

provides DETAIL information plus summarized data on the usage of records and blocks in ENSCRIBE file structures. (See Figure 3-4 for the STATISTICS option listing format.)

PARTONLY

limits the function of STAT[ISTICS] to the specific partitions of a partitioned file named in <fileset-list>. If you omit PARTONLY, STAT[ISTICS] provides information for all partitions of the file.

→

EXTENTS

provides a listing of extent allocation by file. (See Figure 3-5 for the EXTENTS option listing format.)

USER

restricts the display to files in <fileset-list> owned by the user identified by <group-id>,<user-id> or <groupname>.<username>. If you include USER but omit <group-id>,<user-id> and <groupname>.<username>, the display is restricted to files in <fileset-list> owned by the logon user.

Listing Format for INFO Command with No Options

Figure 3-2 shows the listing format for the INFO command with no options.

```
                  CODE  EOF  LAST MODIF  OWNER  RWEP TYPE  REC BLOCK
[\<system-name>.] $<volume-name>.<subvolume-name>
<name> <open-  <code> <eof>   <mod>    <owner><sec><type><rec><block>
    .  state>    .     .       .         .     .     .    .     .
    .     .      .     .       .         .     .     .    .     .
    .     .      .     .       .         .     .     .    .     .
```

Figure 3-2. FUP INFO Command Listing Format (No Options)

The variables that appear in Figure 3-2 are:

<name> is the disc file name of the file whose characteristics are being displayed.

<open-state> is the open state of the file, displayed as:

<null>	File is not open, crashed, or broken.
O	File is open.
OB	FIle is open but has gotten an I/O or consistency check failure and needs a media recovery at some point (DP2 files only).
?	File is crash-open; that is, it was open when a total system failure occurred, or when the volume where it resides became unavailable.
?B	File is crash-open and broken (DP2 files only).
R	File cannot be opened; media recovery is needed.

<code> is the file code. Tandem Computers Incorporated has reserved file codes 100-999 for its own use. Current reserved file codes are listed in Table 3-1.

Letters appearing after the code indicate the following:

"A" means the file is audited by TMF.

"L" means the file is licensed. (See the FUP LICENSE command for details.)

"P" means the PROGID attribute of the file is on. (See the FUP SECURE command for details.)

Note that CODE 0 (zero) is the default code for user-created files; CODE 0 appears as "CODE " in the FUP INFO listing.

<eof> is the number of bytes contained in the file.

<mod> is the date the file was last written. If the file was modified today, the date is blank, and only the time of day is given. Otherwise, the year, month, day, and time are given.

<owner> is the identification number of the file's owner, given as:

<group-id> , <user-id>

The super ID user (255,255) is given as -1.

Table 3-1. System File-Code Definitions
(Continued on next page)

File Code	Definition
100	Object File
101	EDIT-format file
110	VS recovery file
111	VS stack dump file (data area image)
120	Spooler control files
121	Spooler control files (Spooler versions C11 and later)
122	Spooler control files (Spooler versions C11 and later)
130	INSPECT save file
170	XRAYSCAN structured output files for ENFORM reports
223	ENABLE log files
250-299	Reserved for TRANSFER files
	250 Profile file 251 Session file 252 Item descriptor file 253 Recipient file 254 Folder file 255 Item data file 256 Distribution list file 257 Ready file 258 Time file 259 Network file 260 Inverted folder file
300-399	Reserved for TPS files (PATHWAY)
	300 TCL program directory file 301 TCL program code file 302 SCREEN COBOL symbol file 305 TCP data area swap file 306 AM control file 307 Path TCP dump file 308 PATHWAY trace file 309 PATHMON stack dump
400	Tape simulator control file
401	Tape simulator data file
410-419	Reserved for EXERCISE files
	410 EXERCISE message file 411 EXERCISE error information file 430 TANDUMP segmented save file

Table 3-1. System File-Code Definitions (Continued)

File Code	Definition
500	1.5 processor microcode object file
502	1.5 microcode for SHADOW
505	5106 Tri-Density tape drive microcode file
510	Standard (unformatted) microcode file
520	NonStop TXP processor microcode file
600-620	Reserved for MUMPS system files
	600 MUMPS global file 601 MUMPS routine file 602 MUMPS global directory file
660-669	Reserved for ENCORE files
	660 Capture file
888	ENFORM compiled queries

<sec> is the security assigned to the file, given as:

<rwep>

<r> = security level for read
<w> = security level for write
<e> = security level for execute
<p> = security level for purge

Values for <rwep> are:

- = local super ID only
O = owner only (local)
G = member of owner's group (local)
A = any user (local)
U = member of owner's user class--that is, owner only (local or remote)
C = member of owner's community--that is, member of owner's group (local or remote)
N = any user (local or remote)

<type> is the file type of the file. <type> is one or more of these:

<null> = unstructured
R = relative file structure
E = entry-sequenced file structure
K = key-sequenced file structure
<t>A = file has alternate key
P<t> = file is partitioned
XP<t> = file is an extra partition

where <t> can be one or more of R, E, K, P, or X.

<rec> for ENSCRIBE file structures, is the file's logical record length in bytes. This field is blank for unstructured files.

<block> for ENSCRIBE file structures, is the file's block length in bytes. This field is blank for unstructured files.

Example of Standard (No Option) Listing

This example shows the standard FUP INFO listing for three files the current default subvolume and volume ($VOL1.SVOL). The listing gives the following information about the file PARTFILE: PARTFILE is a partitioned, key-sequenced file with an alternate key; its record size is 80 bytes, its block size is 1024 bytes, and it contains 3072 bytes of data; PARTFILE has file code 0, was last modified on August 12, is owned by the user with user ID 8,1, and has security "AO--".

```
-INFO (PARTFILE, MYFILE, ALTFILE)

            CODE  EOF  LAST MODIF    OWNER  RWEP TYPE  REC BLOCK
$VOL1.SVOL
 PARTFILE        3072 14AUG84 14:55 8,1    AO-- PKA    80  1024
 MYFILE          5120 22OCT84 11:44 8,44   AOAO RA     10  1024
 ALTFILE         2048 22OCT84 11:46 8,44   AOAO K      11  1024
```

Listing Format for INFO Command with DETAIL Option

Figure 3-3 shows the listing format used by the FUP INFO command with the DETAIL option.

```
<file-name>                                    <date-and-time>
    TYPE <file-type>
    CODE <file-code>
    EXT ( <pri-num> PAGES, <sec-num> PAGES )
    [ MAXEXTENTS <maximum-extents> ]
    [ BUFFERSIZE <unstructured-buffer-size> ]
    [ REFRESH ]
    [ BUFFERED ]
    [ AUDITCOMPRESS ]
    [ VERIFIEDWRITES ]
    [ SERIALWRITES ]
  { REC <record-length>
    BLOCK  <data-block-length>
 {{ IBLOCK <index-block-length>
    KEYLEN <primary-key-length>
    KEYOFF <primary-key-offset>
  [ DCOMPRESS ] [, ICOMPRESS ]  }}
  [ ALTKEY ( <key-spec>, FILE <alt-fnum>,
             KEYOFF <alt-key-offset>, KEYLEN <alt-key-len>
             [, [NO] UNIQUE ] [, [NO] UPDATE ]
              , NULL <null-value>
              , NO NULL              )
      .
    ALTFILE ( <alt-fnum> , <alt-file-name> )
      .
      .                                          ]  }
  [ PART ( <part-num> , [ \ <system-name>.] $<volume-name>,
           <pri-ext>, <sec-ext> {{ , <partial-key-value> }} )
      .
      .                                 ]
    [ AUDIT ]
    [ ODDUNSTR ]
    OWNER <group-id>,<user-id>
    [ SECONDARY PARTITION ]
    SECURITY (RWEP): <sec> [, PROGID] [, CLEARONPURGE] [, LICENSE]
    MODIF:  <modif> [, <open-state> ]
    EOF <eof>  (<percentage>% USED)
    EXTENTS ALLOCATED: <num-ext>
  { FREE BLOCKS <num-data-blocks> {{, <num-index-blocks>
    INDEX LEVELS: <num-index-levels> }} }
```

Figure 3-3. FUP INFO Command Listing Format (DETAIL Option)

In Figure 3-3, single braces ({ }) enclose parts of the display that appear only for structured files. Double braces ({{ }}) enclose parts of the display that appear only for key-sequenced files. Headers and variables that are described as FUP SET command parameters later in this section are not described here.

Headers and variables that appear in Figure 3-3 are:

<date-and-time> gives the current system date and time when the FUP INFO DETAIL listing was produced.

TYPE <file-type> is given as U for unstructured files, R for relative, E for entry-sequenced, and K for key-sequenced.

CODE <file-code> is given as for the standard (no-option) FUP INFO listing format.

EXT (<pri-num> PAGES, <sec-num> PAGES)

gives the sizes of the first extent allocated (<pri-num> PAGES) and of secondary extents (<sec-num> PAGES).

MAXEXTENTS <maximum-extents>

for DP2 files, indicates the current setting of the maximum number of extents that can be allocated for this file.

BUFFERSIZE <unstructured-buffer-size>

for unstructured DP2 files, indicates the current setting of the default internal transfer size used in file accesses.

REFRESH indicates that the file's disc label is updated whenever its file control block (FCB) changes (for example, when the end-of-file [EOF] marker is changed).

BUFFERED for DP2 files, indicates that writes to the file are buffered by default.

AUDITCOMPRESS for audited or nonaudited DP2 files, indicates that compressed audit-checkpoint messages are generated by default.

VERIFIEDWRITES for DP2 files, indicates that writes to the file are verified by default.

SERIALWRITES	for DP2 files, indicates that serial mirror writes are done by default.
DCOMPRESS	means that data compression was specified for the file.
ICOMPRESS	means that index compression was specified for the file.
ALTKEY	shows information about an alternate-key field; <key-spec> is the <key-specifier> value for an alternate-key field, as described for the SET command in this section.
[NO] UNIQUE	tells whether "key is unique" is set for <key-specifier>. Default is NO UNIQUE.
[NO] UPDATE	tells whether "automatic updating" is set for <key-specifier>. Default is UPDATE.
NULL <null-value> NULL	tells whether a null value is set for <key-specifier>. If a value is specified, <null-value> must be an ASCII character inside quotation marks or an integer in the range from 0 to 255, inclusive. Default is NO NULL. See the ENSCRIBE Programming Manual for information about null values.
ALTFILE	lists information about an alternate-key file.
PART	gives information about a partition.
AUDIT	indicates a file audited by TMF.
ODDUNSTR	indicates an odd unstructured file (one in which input/output operations are not rounded upward to even byte boundaries).

SECONDARY PARTITION

indicates a file that is a secondary partition of a partitioned file.

SECURITY (RWEP): <sec>

indicates the security setting for the file.

PROGID	marks a file whose PROGID bit is set (see the SECURE command in this section and the description of system security in the GUARDIAN Operating System User's Guide).

CLEARONPURGE — marks a file whose CLEARONPURGE bit is set (see the SECURE command).

LICENSE — indicates the file is licensed (see the LICENSE command).

MODIF: <modif> — gives the date and time of the file's last modification.

<open-state> — indicates whether the file is open. This field can be one of:

OPEN	File is open.
OPEN, BROKEN	File is open but has gotten an I/O or consistency check failure and needs media recovery (DP2 files only).
QUESTIONABLE	File is crash-open. That is, the file was open when a total system failure occurred, or the volume where the file resides became unavailable while the file was open.
QUESTIONABLE, BROKEN	File is crash-open and broken (DP2 files only).
RECOVERY NEEDED	File cannot be opened; media recovery is needed. See the Transaction Monitoring Facility (TMF) System Management and Operations Guide for information about recovery.

EOF <eof> (<percentage>% USED)

for unstructured files, is the end-of-file pointer containing the relative byte address of the next byte after the last significant data byte in the file.

For structured files, is the relative byte address of the first byte of the next available block.

<percentage> shows the percentage of available file space currently used. It is calculated as the percentage <eof> would be of all available space if all extents were allocated.

EXTENTS ALLOCATED: <num-ext>

indicates the number of extents allocated for the file.

FREE BLOCKS <num-data-blocks> , <num-index-blocks>
INDEX LEVELS: <num-index-level>

For structured files, <num-data-blocks> indicates the number of free data blocks remaining for the file.

For key-sequenced files, <num-index-blocks> indicates the number of free index blocks used to point to the data blocks.

For key-sequenced files, <num-index-levels> indicates the number of index levels used for index blocks.

Example of DETAIL Option Listing

The example that follows shows the FUP INFO, DETAIL listing for PARTFILE, a key-sequenced, partitioned file with alternate keys:

```
-INFO PARTFILE, DETAIL

$VOL1.SVOL.PARTFILE            10/11/84 14:16
    TYPE K
    EXT ( 1 PAGES, 1 PAGES )
    REC 80
    BLOCK  1024
    IBLOCK 1024
    KEYLEN 10
    KEYOFF 0
    ALTKEY ( "ab", FILE 0, KEYOFF 10, KEYLEN 10 )
    ALTKEY ( "cd", FILE 1, KEYOFF 20, KEYLEN 10 )
    ALTFILE ( 0, $VOL1.SVOL.AK1 )
    ALTFILE ( 1, $VOL1.SVOL.AK2 )
    PART ( 1, $VOL2, 1, 1, "AA" )
    OWNER 8,1
    SECURITY (RWEP): AAAA
    MODIF:  5/11/84 11:20
    EOF 3072  ( 9.4% USED)
    EXTENTS ALLOCATED: 2
    FREE BLOCKS 1
    INDEX LEVELS: 1
```

Listing Format for INFO Command with STATISTICS Option

The INFO command with the STATISTICS option displays the information shown for the DETAIL option (see Figure 3-3) plus the section shown in Figure 3-4. (However, this section is not listed for unstructured or entry-sequenced files.)

```
                TOTAL      TOTAL    AVG #      AVG        AVG %
 LEVEL         BLOCKS      RECS     RECS      SLACK      SLACK      [ PART ]
<level> <t-blocks> <t-recs> <a-recs> <a-slack> <a-%-slack> [<name>]
   .         .         .        .         .          .          .
   .         .         .        .         .          .          .
[ FREE    <t-blocks>           ]
[ FREE                <t-recs> ]
[ BITMAP  <t-blocks> ]
```

Figure 3-4. FUP INFO Command Listing Format (STATISTICS Option)

The variables that appear in Figure 3-4 are:

<level> indicates the tree level of the entry. Values for <level> are:

DATA indicates that the entry is for the data level. This is the only level shown for relative and entry-sequenced files.

A 1 or greater indicates that the entry is for an index level. One (1) is the lowest index level. Index levels are shown only for key-sequenced files.

<t-blocks> is the total number of blocks in use at the indicated level.

<t-recs> is the total number of records at the indicated level. At the DATA level, <t-recs> is the total number of data records in the file.

<a-recs> is the average number of records for each block at the indicated level.

<a-slack> is the average number of unused bytes for each block at the indicated level.

<a-%-slack>	is the average percentage of unused bytes for each block at the indicated level.
<name>	is shown only if the file has extra partitions. <name> is the volume name of the partition associated with the entry.
FREE <t-blocks>	for key-sequenced files, is the total number of unused blocks in the file between the beginning of the file and the current EOF location.
FREE <t-recs>	for relative files, is the total number of empty records in the file between the beginning of the file and the current EOF location.
BITMAP <t-blocks>	for DP2 relative and key-sequenced files only, is the number of bitmap blocks.

Example of STATISTICS Option Listing

This example shows the FUP INFO, STATISTICS listing for a key-sequenced, partitioned file:

```
-INFO PARTFILE,STATISTICS

 (DETAIL option listing displays first, followed by this)

        TOTAL    TOTAL   AVG #    AVG    AVG %
LEVEL   BLOCKS    RECS    RECS   SLACK   SLACK   PART
  1         1       1     1.0     996      97    $VOL1
 DATA       1      12    12.0     338      33
 FREE       1
  1         1       7     7.0     938      92    $VOL2
 DATA       7     117    16.7      77       8
 FREE       1
```

Listing Format for INFO Command with EXTENTS Option

Figure 3-5 shows the listing format used by the FUP INFO command with the EXTENTS option.

```
<file-name>                                  <date-and-time>
    EXTENT      # OF PAGES    STARTING PAGE [   PART   ]
  <extent-no>   <num-pages>   <start-page>   [ <name> ]
```

Figure 3-5. FUP INFO Command Listing Format (EXTENTS Option)

The variables that appear in Figure 3-5 are:

<file-name>	is the name of the file being listed by FUP INFO.
<date-and-time>	are the current system date and time when the listing was produced.
<extent-no>	is the ordinal extent number of the entry. The first extent in a file is designated extent 0. If no extents are allocated, the value is NONE.
<num-pages>	is the number of disc pages (2048-byte units) that compose the indicated extent.
<start-page>	is the absolute page address of the first page of the indicated extent.
<name>	for partitioned files, is the partition name associated with the entry.

Example of EXTENTS Option Listing

This example shows the FUP INFO, EXTENTS listing for PARTFILE, a key-sequenced, partitioned file:

```
-INFO PARTFILE,EXTENTS

$VOL1.SVOL.PARTFILE            11/03/84 14:45
      EXTENT  # OF PAGES  STARTING PAGE  PART
          0            1         12246    $VOL1
          1            1         12247
          0            1           239    $VOL2
          1            1           293
          2            1           294
          3            1           297
          4            1           307
```

LICENSE Command

If you are the super ID user (with user ID 255,255), you can use the LICENSE command to permit nonprivileged users to execute TAL programs containing privileged attributes (CALLABLE or PRIV attributes). Only the super ID user can run a privileged program that is not licensed. Likewise, only a super ID user can license privileged programs or use the REVOKE command to revoke the license of a privileged program. (For more information on licensing programs, see the GUARDIAN Operating System User's Guide.)

The syntax of the LICENSE command is:

```
LICENSE <fileset-list>

<fileset-list>

   is a list of files to be licensed for use by
   nonprivileged users.
```

Consideration

If a licensed, privileged program is opened with write access, the file becomes unlicensed.

Example

After the super ID user enters this command, nonprivileged users on the local system can run the privileged program stored in the disc file MYPROG:

```
-LICENSE MYPROG
```

LISTOPENS Command

Use the LISTOPENS command to list the processes that have open the files you specify in your command. LISTOPENS also gives other related information.

The syntax of the LISTOPENS command is:

```
LISTOPENS [ / OUT <list-file> / ] <fileset-list>
          [ , SCRATCH <scratch-file-name> ]
```

OUT <list-file>

names a file or device to receive the listing output of the LISTOPENS command.

<fileset-list>

is a list of files for which "opens" are to be displayed.

SCRATCH <scratch-file-name>

names a file or volume to be used for temporary storage during the sorting phase. If you omit this option, LISTOPENS uses a temporary file on the default volume.

Listing Format

Figure 3-6 shows the listing format for the FUP LISTOPENS command.

```
 [ \<system-name>.]$<volume-name>.<subvolume-name>.<file-name>
   PID        MODE      USERID     SD  MYTERM    PROGRAM FILE NAME
<s>,<c,p> <-p><a> <-e> <g,u>    <sd> <term>     <prog-name>
          <-b>
```

Figure 3-6. FUP LISTOPENS Command Listing Format

In Figure 3-6, the variables in the first line of the display give the file name you named in your LISTOPENS command.

The other variables that appear in Figure 3-6 are:

<s> is the network system number of the system running the process that has open the file you specify in your command.

<c,p> <-p> <-b> is the process's CPU number and process number. The <-p> or <-b> indicates that this is the primary or backup process (respectively) of a NonStop process pair.

<a> is the access mode:

R	Read
R/W	Read/Write
E	Execute

<-e> is the exclusion mode:

S	Shared
E	Exclusive
P	Protected

<g,u> is the group ID, user ID of the process accessor ID.

<sd> is the sync or receive depth specified by the process when the file was opened. (See the GUARDIAN Operating System Programmer's Guide for a discussion of the file system's automatic path error recovery for disc files and a description of what happens when a file is opened. See the System Procedure Calls Reference Manual for a complete syntax description of the OPEN procedure).

<term>	is the name of the process's home terminal: [\<system-name>.]$<term-name>
<prog-name>	is the program file name of the program that has the specified file open: $<volume-name>.<subvolume-name>.<file-name> appears for a user process. $SYSTEM.SYS<nn>.OSIMAGE is for a system process. ($SYSTEM.SYS<nn> is the subvolume containing the GUARDIAN operating system image currently in use, and <nn> is a two-digit octal integer.)

Example

This example displays a list of all processes that currently have open the file MYFILE (a file in the current default volume and subvolume):

```
-LISTOPENS MYFILE
```

LOAD Command

Use the LOAD command to load data into a structured disc file without affecting any associated alternate-key files. Data that is in the file being loaded is overwritten.

The input records of key-sequenced files can be in sorted or unsorted order; FUP assumes the files are unsorted unless you specify otherwise. For key-sequenced files, you can also specify the percentage of slack space to be left for future insertions.

The syntax of the LOAD command is:

```
LOAD <in-file-name> , <destination-file-name>
     [ , <load-option> ]...
```

<in-file-name>

names the file containing the records to be loaded. This file can be a disc file, a nondisc device, or a process.

<destination-file-name>

specifies an existing disc file in which the records from <in-file-name> are to be loaded.

<load-option>

for key-sequenced destination files, indicates whether the <in-file-name> records are in sorted order (and thus whether a sort should be performed) and specifies load options; for all destination files, specifies the <in-file-name> format. <load-option> is any one of these:

```
SORTED
<in-option>
PAD <pad-character>
PARTOF $<volume-name>
<key-seq-option>
```

→

SORTED

for a key-sequenced destination file only, specifies that the records in <in-file-name> are in the destination file's key-field order; therefore a sort of the <in-file-name> records is not performed. If you omit this option, FUP sorts the <in-file-name> records before loading the <destination-file-name>.

<in-option>

specifies the format and control of the <in-file-name>. <in-option> is any one of these:

```
RECIN <in-record-length>
BLOCKIN <in-block-length>
TRIM <trim-character>
VARIN
EBCDICIN
SHARE
[ NO ] UNLOADIN
[ NO ] REWINDIN
SKIPIN <num-eofs>
REELS <num-reels>
```

For a description of these options, see the description of <in-option> for the COPY command.

PAD <pad-character>

means that records containing fewer than <in-record-length> bytes are padded with <pad-character> up to the record length specified in the file label. Specify <pad-character> as:

```
"c"
```

(a single ASCII character inside quotation marks) or:

```
{ 0 : 255 }
```

(an integer in the range from 0 to 255, inclusive, specifying a byte value).

→

If you omit <pad-character>, FUP uses an ASCII null character.

PARTOF $<volume-name>

for key-sequenced, partitioned files only, loads only the partition named in <destination-file-name>; $<volume-name> is the volume containing the primary partition of the destination file. (See "Considerations.")

<key-seq-option>

is an option pertaining to loading key-sequenced files. <key-seq-option> is any one of these:

MAX <num-records>
SCRATCH <scratch-file-name>
SLACK <percentage>
DSLACK <percentage>
ISLACK <percentage>

(The next two options pertain to sorting the <in-file-name> records. If you specify SORTED for <load-option>, you can ignore these options.)

MAX <num-records>

is the number of records in <in-file-name>. Specify <num-records> as an integer in the range from 0 to 2,147,483,647, inclusive. This value need not be exact, but it should be equal to or greater than the actual number of records in <in-file-name>. FUP uses <num-records> to determine the size of the scratch file used by the SORT process. The default <num-records> is 10,000.

→

SCRATCH <scratch-file-name>

names a file or volume to be used for temporary storage during the sorting phase. If you omit this option, FUP uses a scratch file on the default volume.

(The next three options specify the minimum percentage of space to be left in index blocks and in data blocks for future insertions. If space is not available when an insertion is made, a block split occurs.)

SLACK <percentage>

sets the minimum percentage of slack space in both index and data blocks. Specify <percentage> as a value in the range from 0 to 99, inclusive. If you omit this option, FUP leaves no slack space.

DSLACK <percentage>

sets the minimum percentage of slack space in data blocks. If you omit this option, FUP uses the SLACK percentage value.

ISLACK <percentage>

sets the minimum percentage of slack space in index blocks. If you omit this option, FUP uses the SLACK percentage value.

Considerations

- If you specify the PARTOF option when loading a nonpartitioned file, you receive an error message (see the System Messages Manual).
- When loading partitioned files, consider these restrictions:

  --In a partitioned file, the range of keys for the different partitions is stored in the primary partition.

--If you attempt to load a secondary partition when you have not specified the PARTOF option, you receive an error message (see the System Messages Manual).

--If you do not specify PARTOF and if <destination-file-name> is the primary partition, FUP assumes that all partitions are to be loaded.

--To load a secondary partition, you must specify the name of the secondary partition as <destination-file-name> and specify the name of the volume where the primary partition resides for the PARTOF option.

--To load the primary partition only, specify the name of the primary partition as <destination-file-name>, and specify the name of the primary volume for the PARTOF option.

- If the <in-file-name> records must be sorted, then disc space for the sort scratch file and for <destination-file-name> must exist concurrently during the sorting phase.

- Be careful when using the PAD and TRIM options in a FUP LOAD or COPY operation. If your data contains the <trim-character> or <pad-character>, data might be altered or lost. For example, suppose you pad each record in a data file with zeros to a standard size in bytes and then store the records in another file. If you later trim the trailing zeros when you FUP LOAD or COPY the stored records, any original data ending with zero is trimmed. To avoid this problem, use a <pad-character> or <trim-character> that is not contained in your data.

Example

This example loads data from the tape device $TAPE into the key-sequenced disc file KSFILE. Records to be loaded from $TAPE are already sorted in the order of KSFILE's key fields. The minimum percentage of slack space to be left in data blocks is 10 percent:

```
-LOAD $TAPE, KSFILE, SORTED, DSLACK 10
```

LOADALTFILE Command

Use the LOADALTFILE command to generate from a primary file the alternate-key records for a designated alternate-key file and to load those records into that file. You can also specify the amount of slack space reserved for future insertions.

The syntax of the LOADALTFILE command is:

```
LOADALTFILE <key-file-number> , <primary-file-name>
            [ , <key-seq-option> ]...
```

<key-file-number>

selects the alternate-key file to be loaded. Specify <key-file-number> as an integer in the range from 0 to 255, inclusive, indicating an alternate-key file of the primary file. The alternate-key file must already exist. You set the key-file number with the FUP SET command; you can display the key-file number with the FUP INFO command.

<primary-file-name>

names the existing primary file whose alternate-key records are to be generated and loaded into the file indicated by <key-file-number>. Partial file names are expanded using the current default system, volume, and subvolume names where necessary.

<key-seq-option>

specifies options for loading the alternate-key file. <key-seq-option> is any one of these:

```
MAX <num-records>
SCRATCH <scratch-file-name>
SLACK <percentage>
DSLACK <percentage>
ISLACK percentage
```

→

(The next two options pertain to sorting the alternate-key records generated from the primary file.)

MAX <num-records>

specifies the number of records to be read from <primary-file-name>. If you omit this option, 10,000 is used. Specify <num-records> as a value in the range from 0 to 2,147,483,647, inclusive. FUP multiplies <num-records> by the number of alternate keys associated with <key-file-number> to determine the size of the scratch file used by the SORT process. The number of records you specify need not be exact, but if you include <num-records>, its value should be equal to or greater than the actual number of records in the source file.

SCRATCH <scratch-file-name>

specifies a file or volume to be used for temporary storage during the sorting phase. If you omit this option, FUP uses a scratch file on the default volume.

The following options specify the minimum percentage of space to be left in index and data blocks for future insertions. If space is not available when an insertion is made, a block split occurs.

SLACK <percentage>
DSLACK <percentage>
ISLACK <percentage>

For a description of these options, see the description of the LOAD command in this section.

Considerations

- LOADALTFILE ignores any NO UPDATE specifications.
- LOADALTFILE honors any NULL specification defined for a key field (that is, an alternate-key record is not generated for a field consisting solely of null characters if such a null character was defined).
- When you execute LOADALTFILE, a sort operation is performed. The primary file is read sequentially by way of its primary-key field. For each record read from the primary file, the alternate-key records are generated and written to the SORT process. When the sort completes, the sorted records are read from the SORT process and loaded into the indicated alternate-key file. Note that during the sorting phase, disc space for the sort scratch file and for the alternate-key file must exist concurrently.

Example

This command generates alternate-key records from the primary file KSFILE and loads those records into the alternate-key file with file number 0. A minimum of 10 percent slack space is left in the data blocks of the alternate-key file:

```
-LOADALTFILE 0, KSFILE, DSLACK 10
```

PURGE Command

Use the PURGE command to delete a single disc file, a set of files, or several sets of files.

The syntax of the PURGE command is:

```
PURGE [ ! ]  <fileset> [ , <fileset> ] ...  [ ! ]
```

!

in either or both of the indicated positions, means purge the files without prompting you for permission. If you do not include "!", FUP asks you if each file in <fileset> should be purged (the prompt is "PURGE?").

<fileset>

names the file or set of files to be purged. Partial file names are expanded using the current default system, volume, and subvolume where necessary.

If you include "!" in your command, FUP purges the files in <fileset> without requesting your permission.

If you do not include "!", the FUP INFO listing is displayed for each file in <fileset>, followed by the prompt "PURGE?". To purge the file, enter "Y" or "y". If you enter any other response, the file is not purged.

Considerations

- If you try to purge a file that is audited by the Transaction Monitoring Facility and for which transaction-mode record or file locks are pending, your purge request fails and file-system error 12 is returned (whether or not the processes that opened the file still exist).

- You can purge a file only if it is not currently in use (not open, running, or being backed up). In addition, you must have purge access to the file or be the super ID user.

- If you purge a file for which you previously specified the FUP SECURE option CLEARONPURGE, the disc process physically deletes the file's data from the disc (overwrites it with blank data) before deleting the file name from the directory.

Examples

This example purges the file WKLYRPRT in the current default subvolume without prompting you for permission:

```
-PURGE WKLYRPRT !
1 FILE PURGED
```

This command displays the FUP INFO listing for each file in the current default subvolume and prompts you for permission to purge each file:

```
-PURGE *
```

PURGEDATA Command

Use the PURGEDATA command to remove all data from a file. PURGEDATA does not physically purge data. Instead, it purges data "logically" by setting the end-of-file pointer to the beginning of the file and resetting other values in the file label to those of an empty file. The file still exists, and its file name can be displayed with the FILES command. Disc space previously allocated to the purged file can now be allocated to other files, or to the original file.

The syntax of the PURGEDATA command is:

```
PURGEDATA <fileset-list> [ , PARTONLY ]
```

<fileset-list>

is a list of files whose data is to be purged. Partial file names are expanded using the current default system, volume, and subvolume where necessary.

PARTONLY

purges data in any primary or secondary extents of partitioned files that reside in <fileset-list>. If you omit PARTONLY, data is purged from all partitions of partitioned files, but only if the primary partitions of the files reside in <fileset-list>.

Considerations

- You must have both write and purge access to a file (or be the super ID user) in order to use the PURGEDATA command to purge data from that file.
- If you try to use the PURGEDATA command on a file whose AUDIT option is set (for auditing by the Transaction Monitoring Facility); your purge request fails, and file-system error 80 is returned.

- For a PURGEDATA operation, FUP attempts to open the specified file with exclusive access. Therefore, your PURGEDATA command fails if the file cannot be opened with exclusive access (for example, if the file is already open).

- The CLEARONPURGE option set with the FUP SECURE command has no effect on the PURGEDATA command. After the PURGEDATA command is executed, the data is physically present on the disc (but inaccessible) until it is overwritten by new data.

Example

This example logically purges the data from MYFILE in the current default subvolume:

```
-PURGEDATA MYFILE
```

RENAME Command

Use the RENAME command to change the file name or subvolume name of a disc file or to rename sets of files. You can also use RENAME to "reorganize" your files by renaming one or more files in one or more subvolumes to a single subvolume. If the destination subvolume does not already exist, it is created during the renaming process. RENAME cannot, however, transfer files from one volume to another. Use FUP DUP or FUP COPY for that purpose.

The syntax of the RENAME command is:

```
RENAME <old-fileset-list> , <new-fileset> [ , PARTONLY ]
```

<old-fileset-list>

is a <fileset-list> specifying the files to be renamed. Partial file names are expanded using the current default system, volume, and subvolume where necessary. You can use "*" to indicate all the files in the specified or current default subvolume.

<new-fileset>

is a <fileset> specifying the new names of the files. To rename more than one file, you must specify the disc file name portion of the <new-fileset> as "*"; then each file is given the disc file name of its corresponding input file, and the new subvolume name is taken from the <new-fileset> specification.

PARTONLY

for partitioned files, specifies that only partitions included in <old-fileset-list> are to be renamed. If you omit PARTONLY, FUP renames all partitions of partitioned files if their primary partitions reside in <old-fileset-list>. PARTONLY has no effect on files that are not partitioned.

Considerations

- If you try to rename a file whose AUDIT option is set (for auditing by the Transaction Monitoring Facility), the rename request fails and file-system error 80 is returned.
- You cannot rename a file that is currently open with exclusive access.
- You must have purge access to the file or be the super ID user to rename a file. However, to make any changes to a file, you must also have read access to the file.

Example

This example changes the subvolume name of all the files in the current default subvolume and all the files in the SVOL subvolume. All the files retain their original file names but are renamed to the subvolume NEWSVOL:

```
-RENAME (SVOL.*, *), NEWSVOL.*
```

RESET Command

Use the RESET command to restore one or more file-creation attributes to the default settings. (See the FUP SET command for a list of file-creation attributes and their default values.)

The syntax of the RESET command is:

```
RESET [ <create-spec> [ , <create-spec> ]... ]

<create-spec> is any one of these:

   TYPE
   CODE
   EXT
   REFRESH
   AUDIT
   REC
   BLOCK
   IBLOCK
   COMPRESS
   DCOMPRESS
   ICOMPRESS
   KEYLEN
   KEYOFF
   ALTKEY [ <key-specifier> ]
   ALTKEYS
   ALTFILE [ <key-file-number> ]
   ALTFILES
   ALTCREATE
   PART [ <partition-num> ]
   PARTS
   PARTONLY
   MAXEXTENTS
   BUFFERSIZE
   BUFFERED
   AUDITCOMPRESS
   VERIFIEDWRITES
   SERIALWRITES
   ODDUNSTR
```

Considerations

- Each <create-spec> keyword in a RESET command resets the corresponding creation attribute to its default setting. If you do not include any <create-spec> in your RESET command, FUP resets all creation attributes to their default settings.
- If you specify the plural keywords ALTKEYS, ALTFILES, or PARTS, FUP resets all creation values that pertain to alternate keys, to alternate-key files, or to secondary partitions, respectively. (See the SET command for the default values.)

Examples

Suppose you had set the file-creation attributes to create a relative structured DP2 file (named $COMPUTER.BOOKS.PASCAL) with a record size of 10 bytes and with an alternate-key file (named $COMPUTER.BOOKS.BLAISE). The SHOW command displays:

```
TYPE R
EXT ( 1 PAGES, 1 PAGES )
REC 10
BLOCK 1024
ALTKEY ("aa", FILE 0, KEYOFF 0, KEYLEN 5)
ALTFILE (0, $COMPUTER.BOOKS.BLAISE)
ALTCREATE
MAXEXTENTS 16
```

To reset the record size file-creation attribute to the default value of 80 bytes, enter:

```
-RESET REC
```

Now the FUP SHOW display includes this line:

```
REC 80
```

A single RESET command with no <create-spec> restores all the defaults (this display includes two DP2 file attributes--MAXEXTENTS and BUFFERSIZE):

```
-RESET
-SHOW
   TYPE U
   EXT ( 1 PAGES, 1 PAGES )
   MAXEXTENTS 16
   BUFFERSIZE 4096
```

REVOKE Command

The ordinary user or the super ID user (with user ID 255,255) can use the REVOKE command in these situations:

- By including no options, the super ID user can revoke the license for a privileged program file named in the <fileset-list> (see the LICENSE command).
- By including options, any user can reset certain security attributes of the files in the <fileset-list> (see the <secure-option> in the following syntax description).

Only a super ID user can revoke a program's license. Other users can revoke the PROGID or CLEARONPURGE attributes of files they own.

The syntax of the REVOKE command is:

```
REVOKE <fileset-list> [ , <secure-option> ]...
```

<fileset-list>

is a list of files whose licenses or other attributes are to be revoked. Partial file names are expanded using the current default system, volume, and subvolume where necessary.

<secure-option>

is one of three options that can be set by a program or by a SECURE command (see the FUP SECURE command for more information). The three options are:

```
PROGID
PARTONLY
CLEARONPURGE
```

→

PROGID

for program files only, sets the process's accessor ID to the program file's owner ID when the program file is run. Including PROGID in a REVOKE command revokes the PROGID option for the file, so that the process's accessor ID is set to the ID of the process's creator when the program is run.

PARTONLY

for partitioned files, means that only the designated partition is affected by this REVOKE command. If you omit PARTONLY, every partition is affected. If PARTONLY is the only <secure-option> in a REVOKE command, the license for that partition is revoked.

CLEARONPURGE

when set, physically deletes all data within the <fileset-list> from the disc (by overwriting the file space with blank data) when the file is purged. Including CLEARONPURGE in a REVOKE command revokes the CLEARONPURGE option for the file. When you purge a file that does not have CLEARONPURGE set, the disc space is logically deallocated. (This option has no effect on a PURGEDATA operation.)

Consideration

Use the PARTONLY option with REVOKE in these situations:

- To revoke an attribute of a secondary partition named in <fileset-list>; otherwise, your REVOKE command fails with file-system error 72 (attempt to access an unmounted partition)
- To revoke an attribute of the primary partition only; otherwise, FUP revokes the attribute for all partitions of the file

Examples

If you own the partitioned file PARTFILE (in the current default volume and subvolume), entering this command revokes the CLEARONPURGE attribute for the primary partition only:

```
-REVOKE PARTFILE, PARTONLY, CLEARONPURGE
```

The owner of the file or the super ID user can revoke the license of MYPROG in subvolume $SYSTEM.SYSTEM by entering:

```
-REVOKE $SYSTEM.SYSTEM.MYPROG
```

SECURE Command

Use the SECURE command to set or change a file's security attributes. Only the file's owner or the super ID user can execute the SECURE command for any given file.

In a TAL program, you can perform the functions of the SECURE command using a SETMODE call with <function> equal to 1. See the System Procedure Calls Reference Manual for a description of the SETMODE file-system procedure.

The syntax of the SECURE command is:

```
SECURE <fileset-list> [ , [ <security> ]
       [ , secure-option ]... ]

<security> is  { "<security-string>" }
               {  <security-num>     }
```

"<security-string>"

sets new file security for the files in <fileset-list>. Specify <security-string> as a four-character string inside quotation marks.

The four characters in <security-string> assign new values for file security, as follows:

<rwep>

- <r> = security level allowed for read
- <w> = security level allowed for write
- <e> = security level allowed for execute
- <p> = security level allowed for purge

You can use these characters in <security-string>:

- \- = local super ID only (not allowed in <r> position)
- O = owner only
- G = group member or owner
- A = any local user

→

U = any member of owner's user class
(local or remote users with the same group ID and user ID as the file's owner)
C = any member of owner's community
(local or remote users with the same group ID as the file's owner)
N = any local or remote user

<security-num>

is an integer encoding of the file's security. You specify <security-num> in octal notation as:

%ijkkkk

i = 1 if the PROGID option is to be set; 0 if not.

j = 4 if CLEARONPURGE is to be set; 0 if not.

(See the SETMODE procedures in the System Procedure Calls Reference Manual.) Also see the descriptions for the PROGID and CLEARONPURGE options in this syntax description.

kkkk sets the values for read, write, execute, and purge, respectively; k can be:

0 = any local user
1 = member of owner's group
2 = owner
4 = any local or remote user
5 = member of owner's community
6 = member of owner's user class
7 = local super ID user only

Using <security-num>, a standard user can secure a file so that only the super ID user has access (%7777). If you are a standard user, note that specifying "%7777" means that you do not have any access to the file yourself. (You have to ask the super ID user to give you access to the file.)

→

CAUTION

If you omit both <security-string> and <security-num>, SECURE does not set the file's security to your default security but to "O---" security. That is, the file's owner has read access, but only the local super ID user has write, execute, and purge access to the file.

<secure-option> is any one of these three:

PROGID

is the program ID for program files only. It sets the process's accessor ID to the program file's owner ID when the program file is run.

PARTONLY

for partitioned files, changes security only for the designated partition. If you omit PARTONLY, security for every partition of the file is affected, provided the primary partition of the file is included in <fileset-list>; if the primary partition is not included in <fileset-list>, no partitions of the file are affected.

CLEARONPURGE

physically deletes all data in the <fileset-list> from the disc (by overwriting the file space with blank data) when the file is purged. When you purge a file that does not have CLEARONPURGE set, the disc space is logically deallocated; the data is not physically destroyed. (This option has no effect on a PURGEDATA operation.)

Considerations

- You cannot secure a file that is open with exclusive exclusion mode access.
- If a process has a file open when you SECURE the file, the access rights of the process are not affected until the process closes that file.

Examples

To change the security for MYFILE (a file in the current default subvolume) so that any local user can read the file, but only the file's owner can write, execute, or purge the file, enter:

```
-SECURE MYFILE, "AOOO"
```

Entering this command allows any local user to read, write, execute, or purge MYFILE:

```
-SECURE MYFILE, 0
```

This example secures MYPROG so that only the owner can read, write, and purge it, but any local user can execute it. This command also sets the PROGID bit so that the owner ID of MYPROG is used as the process accessor ID when the program is run:

```
-SECURE MYPROG, "OOAO", PROGID
```

To set network security for the local file $OFFICE.BILLS.PAPER so that any local or remote user can read the file, only the local group members can execute the file, and only the local owner can write or purge the file, enter:

```
-SECURE $OFFICE.BILLS.PAPER, "NOGO"
```

SET Command

Use the SET command to change one or more file-creation attributes before you create files. You can specify parameter values explicitly or set them to match those of an existing file.

To display the current file-creation attributes (the values for <create-param>), use the SHOW command. Also, you can restore <create-param> to the default settings with the RESET command.

The syntax of the SET command is:

```
SET <create-param> [ , <create-param> ]...

<create-param> is any one of these:

   LIKE <file-name>
   TYPE <file-type>
   CODE <file-code>
   EXT  {<extent-size>                                  }
        {( <pri-extent-size> , <sec-extent-size> )      }
   [ NO ] REFRESH
   [ NO ] AUDIT
   REC <record-length>
   BLOCK <data-block-length>
   IBLOCK <index-block-length>
   [ NO ] COMPRESS
   [ NO ] DCOMPRESS
   [ NO ] ICOMPRESS
   KEYLEN <key-length>
   KEYOFF <key-offset>
   ALTKEY ( <key-specifier> { , <altkey-param> }... )

      <altkey-param> is any one of these:

         FILE <key-file-number>
         KEYOFF <key-offset>
         KEYLEN <key-length>
         NULL <null-value>
         NO NULL
         [ NO ] UPDATE
         [ NO ] UNIQUE
```

→

```
ALTFILE ( <key-file-number> , <file-name> )
[ NO ] ALTCREATE
PART ( <sec-partition-num>
       , [\<system-name>.]$<volume-name>
       [ , <pri-extent-size> [ , [ <sec-extent-size> ]
       [ , <partial-key-value> ] ] ] )
[ NO ] PARTONLY
MAXEXTENTS <maximum-extents>
BUFFERSIZE <unstructured-buffer-size>
[ NO ] BUFFERED
[ NO ] AUDITCOMPRESS
[ NO ] VERIFIEDWRITES
[ NO ] SERIALWRITES
ODDUNSTR
```

Parameters for All File Types

LIKE <file-name>

sets file-creation attributes to match those of an existing file. FUP expands a partial file name using the current default values for system, volume, and subvolume where necessary.

If you specify a secondary partition of a partitioned file in <file-name>, FUP automatically sets the PARTONLY <create-param> (see PARTONLY option in this syntax description).

TYPE <file-type>

sets the file type. Legitimate values for <file-type> and their meanings are:

- U or 0 = unstructured file
- R or 1 = relative file
- E or 2 = entry-sequenced file
- K or 3 = key-sequenced file

The default setting for <file-type> is U.

→

CODE <file-code>

sets the file code. Specify <file-code> as an integer in the range from 0 to 65,535, inclusive. The file codes 100-999 are reserved for use by Tandem Computers Incorporated. The default setting for <file-code> is 0. Table 3-1 (in the description of the INFO command) lists the reserved file codes.

EXT { <extent-size> | (<pri-extent-size> , <sec-extent-size>) }

sets the extent size. See "Considerations" for an explanation of how DP2 file extents are rounded upward. For structured files, extents should be at least as large as the block size; otherwise, multiple extents are allocated every time a block is required. The default value for <extent-size>, <pri-exent-size>, and <sec-exent-size> is 1 page (2048 bytes).

You can specify the following values for <extent-size>, <pri-extent-size>, and <sec-extent-size>:

0:65535 [PAGE[S]]

specifies the extent size in pages (2048-byte units). The minimum extent size is one page; therefore, specifying 0 extents allocates one page (2048 bytes).

0:134215680 BYTE[S]

specifies the extent size in bytes. FUP rounds upward to the next full page. For example, if you specify 2047 bytes, FUP allocates one page; for 2049 bytes, FUP allocates two pages, and so on.

→

0:134215680 REC[S]

specifies the extent size based on the current settings for <record-length> (REC), <data-block-length> (BLOCK), <index-block-length> (IBLOCK), key-field lengths, and compression settings. FUP rounds upward to the next full page.

0:134 MEGABYTE[S]

specifies extent sizes in million-byte units. FUP rounds the specification upward to the next full page.

[NO] REFRESH

controls whether the file's label is automatically copied to disc whenever the file's file control block is marked as "dirty" (for example, when the end-of-file value changes). The default setting is NO REFRESH.

[NO] AUDIT

designates the file as audited or nonaudited by the Transaction Monitoring Facility. The default is NO AUDIT.

Parameters for All Structured Files

REC <record-length>

sets the record length. For relative and entry-sequenced files, specify <record-length> as an integer in the range from 1 to 4072, inclusive; for DP1 key-sequenced files, the range is from 1 to 2035, inclusive; for DP2 key-sequenced files, the range is from 1 to 4062, inclusive. The default setting for <record-length> is 80 bytes. When data compression is used, the maximum record length is reduced by 1 byte.

→

BLOCK <data-block-length>

sets the data-block length. Specify <data-block-length> as an integer in the range from 1 to 4096, inclusive. The default <data-block-length> is 1024 bytes. Blocks should be no larger than extents; otherwise, multiple extents are allocated every time a block is required.

NOTE

When you are setting file-creation attributes, you can SET BLOCK <data-block-length> to any value within the valid range for this parameter, but FUP rounds this value upward to one of these sizes: 512, 1024, 1536, 2048, 3072, 3584, or 4096 bytes. If you later create a DP2 file with that BLOCK setting, the disc process rounds the block size upward to a valid DP2 block size: 512, 1024, 2048, or 4096 bytes.

Parameters for Key-Sequenced Files

IBLOCK <index-block-length>

sets the index-block length. Give <index-block-length> as an integer in the range from 0 to 4096, inclusive. The default <index-block-length> is 1024 bytes. For DP1 files, FUP rounds IBLOCK settings upward as described for the BLOCK parameter. With DP2 files, however, the index-block size equals the data-block size; FUP uses the data-block size as the index-block size and ignores any IBLOCK specification you make.

[NO] COMPRESS

sets or clears the states of both index compression and data compression. The default setting is NO COMPRESS. For data compression, the key offset must be 0, and the maximum record size is reduced by 1 byte.

→

[NO] DCOMPRESS

sets or clears data compression. The default setting is NO DCOMPRESS. The key offset must be 0 for data compression; data compression lowers the maximum record size by 1 byte.

[NO] ICOMPRESS

sets or clears index compression. The default setting is NO ICOMPRESS.

KEYLEN <key-length>

sets the primary-key length. Specify <key-length> as an integer in the range from 1 to 255, inclusive. To create key-sequenced file structures, you must specify KEYLEN; otherwise, your creation attempt fails.

KEYOFF <key-offset>

sets the primary-key offset. Specify <key-offset> as an integer in the range from 0 to 2034, inclusive. Default setting for <key-offset> is 0.

Parameters for Files Having Alternate-Key Fields

ALTKEY (<key-specifier> { , <altkey-param> }...)

sets an alternate-key specification. You must specify each alternate key separately.

<key-specifier>

is a two-byte value that uniquely identifies this alternate-key field. Specify <key-specifier> as either:

"[c1]c2"

→

(a one- or two-character string in quotation marks)
or:

{ -32,768 : 32,767 }

(an integer between -32,768 and 32,767, inclusive).

You can use any characters for <key-specifier> except 0. If you omit c1, then c1 is treated as a 0.

<altkey-param> is any one of these:

FILE <key-file-number>
KEYOFF <key-offset>
KEYLEN <key-length>
NULL <null-value>
NO NULL
[NO] UPDATE
[NO] UNIQUE

FILE <key-file-number>

sets the key-file number for <key-specifier>. Specify <key-file-number> as an integer in the range from 0 to 255, inclusive. This number is related to an actual file by the ALTFILE <create-param>. Default <key-file-number> is 0.

KEYOFF <key-offset>

sets the key offset for <key-specifier>. The default setting for <key-offset> is 0.

KEYLEN <key-length>

sets the key length for <key-specifier>. You must specify a KEYLEN to create a key-sequenced file; otherwise, your creation attempt fails.

→

NULL <null-value>
NO NULL

specify whether a null value is set for <key-specifier>. If a value is specified, <null-value> must be an ASCII character in quotation marks or an integer in the range from 0 to 255, inclusive. The default is NO NULL. See the ENSCRIBE Programming Manual for an explanation of null values.

[NO] UPDATE

specifies whether or not automatic updating is set for the alternate-key file represented by <key-specifier>. Default is UPDATE.

[NO] UNIQUE

specifies whether or not "key is unique" is set for <key-specifier>. Default is NO UNIQUE.

ALTFILE (<key-file-number> , <file-name>)

gives the file name of an alternate-key file. You must specify ALTFILE for each different <key-file-number> in an ALTKEY specification.

Specify <key-file-number> as an integer in the range from 0 to 255, inclusive, that designates the alternate-key file you are naming. For <file-name>, give the name of the alternate-key file for that <key-file-number>. The primary file and all its alternate-key files must be on volumes of the same disc-process type (that is, DP1 or DP2). FUP expands a partial file name using the current default system, volume, and subvolume names.

If you assign <key-file-number> nonsequentially, FUP renumbers the ALTFILE and ALTKEY attributes at file-creation time to use sequential numbering.

→

[NO] ALTCREATE

sets or clears automatic alternate-key file creation. If you specify ALTCREATE, the alternate-key files are created when you create the primary file. The default setting is ALTCREATE.

Parameters for Partitioned Files

```
PART ( <sec-partition-num>
     , [\<system-name>.]$<volume-name>
       [ , <pri-extent-size> [ , [ <sec-extent-size> ]
       [ , <partial-key-value> ] ] ] )
```

sets secondary partition specifications for partitioned files. Specify each secondary partition separately. All partitions of a file must be on volumes of the same disc-process type (that is, DP1 or DP2).

<sec-partition-num> , \<system-name>.$<volume-name>

names the volume where this secondary partition is to reside. Specify <sec-partition-num> as an integer in the range from 1 to 15, inclusive, to designate the secondary partition. Specify \<system-name> and $<volume-name> as the names of the system and volume to contain the partition. (You specify the file name and the volume of the primary partition when you create the file.)

<pri-extent-size> , <sec-extent-size>

set the primary and secondary extent sizes. Specify any of the values defined for the EXT parameter earlier in this syntax description.

→

<partial-key-value>

for key-sequenced files only, specifies the lowest key value that can reside in this partition. Specify <partial-key-value> as:

"c1c2...cn"

(a string of characters in quotation marks) or:

```
{"c"          }  [, "c"          ]  ...
{ 0 : 255 }     [, { 0 : 255 }]
```

(a list of single characters, each inside quotation marks and separated by commas, and/or integers representing byte values in the range from 0 to 255, inclusive).

You must include <partial-key-value> for each partition of a key-sequenced file. (For the primary partition, the partial-key value is 0.)

[NO] PARTONLY

specifies whether subsequent file creations are to create all partitions of a partitioned file (NO PARTONLY) or to create a single partition (PARTONLY). The default setting is NO PARTONLY.

If you specify PARTONLY when a PART specification is in effect, any file you create is designated as a primary partition. Conversely, if no PART specification is in effect, any file you create is designated as a secondary partition. In either case, you specify the file name of the (primary or secondary) partition when you create the file.

→

Parameters for DP2 Files

MAXEXTENTS <maximum-extents>

sets the maximum number of extents the file can have. Specify <maximum-extents> as an integer in the range from 1 to n, where n is a maximum value determined by the amount of free space left in the file label. The default value for <maximum-extents> is 16, and the absolute maximum you can specify is 978. For partitioned files, <maximum-extents> is always 16. (See the ENSCRIBE Programming Manual for more details.)

BUFFERSIZE <unstructured-buffer-size>

for unstructured files, specifies the size in bytes of the internal buffer used when accessing the file. Specify <unstructured-buffer-size> as an integer in the range from 1 to 4096, inclusive, designating the buffer size in bytes. The default value is 4096 bytes.

The extent size of unstructured DP2 files must be an integral multiple of the BUFFERSIZE (for details, see the description of the EXT option in this syntax description). Also, FUP rounds the buffer size upward to the next valid DP2 block size: 512, 1024, 2048, or 4096 bytes.

[NO] BUFFERED

sets the mode of handling write requests. If you select BUFFERED, then write requests are, by default, buffered in the disc-process cache rather than forced to disc at each request. The default value is BUFFERED for audited files, NO BUFFERED for nonaudited files.

[NO] AUDITCOMPRESS

for audited files, specifies whether the mode of auditing is by generating entire before-after messages, or compressed before-after messages. The default value is NO AUDITCOMPRESS.

[NO] VERIFIEDWRITES

sets the mode for disc writes (that is, verified or not). The default is NO VERIFIEDWRITES.

[NO] SERIALWRITES

sets the value for the open file attribute for selecting serial or parallel mirror writes. The default is NO SERIALWRITES (that is, parallel mirror writes).

Parameter for Odd Unstructured Files

ODDUNSTR

ENSCRIBE unstructured files can exist in two ways--as even unstructured or as odd unstructured. In even unstructured files, any odd byte count you give for reading, writing, or positioning is rounded upward. (This is the default for unstructured files.) Odd unstructured files have no upward rounding; you always read, write, or position at the byte count you give.

To eliminate upward rounding, you must specify ODDUNSTR.

Considerations

- The SET command sets the file-creation attributes only for the current session of FUP. Every time you start a new FUP (which happens when you type FUP at the COMINT prompt), the attributes are reset to their default values. To use the SET and SHOW features, you must use FUP in the interactive mode.

- The REFRESH, DCOMPRESS, ICOMPRESS, and COMPRESS attributes are not passed to alternate-key files when you create the primary-key file. To set these attributes, you must include the NO ALTCREATE option and create the alternate-key files in a separate operation.

- When you create a file, the system rounds the extent size upward according to the following rules:

  --For DP1 files, the extent size of structured files must be greater than or equal to the block size. For key-sequenced files, the extent size must be greater than or equal to the data-block size or the index-block size, whicher is greater.

  --For DP2 structured files, the extent size must be a multiple of the block size; for DP2 unstructured files, the extent size must be a multiple of the unstructured buffer size.

  The following listing summarizes the upward rounding that can occur at file creation. Block and buffer sizes are given in bytes, and extent sizes are given in pages (the default extent size is 1 page). Note that although you can specify different sizes for primary and secondary extents, equal sizes are shown here for simplicity.

Type of File	EXT Setting	Extent Size Created
DP1 files:		
Unstructured file	default	(1, 1)
Unstructured file	any N	(N, N)
Structured files		
with BLOCK <= 2048	default	(1, 1)
with BLOCK <= 2048	any N	(N, N)
with BLOCK > 2048	default	(2, 2)
with BLOCK > 2048	any N	(N, N)
DP2 files:		
Unstructured file		
with BUFFERSIZE <= 2048	any N	(N, N)
with BUFFERSIZE > 2048	even N	(N, N)
with BUFFERSIZE > 2048	odd N	(N+1, N+1)
with BUFFERSIZE > 2048	default	(2, 2)

Structured files		
with BLOCK <= 2048	default	(1, 1)
with BLOCK <= 2048	any N	(N, N)
with BLOCK > 2048	even N	(N, N)
with BLOCK > 2048	odd N	(N+1, N+1)

For a full explanation of extent-size rounding, see the ENSCRIBE Programming Manual.

Examples

To set file-creation attributes for a key-sequenced file with 50-byte records and a primary-key length of 36 bytes, enter:

```
-SET TYPE K, REC 50, KEYLEN 36
```

If you set the BUFFERSIZE for an unstructured DP2 file, FUP rounds the buffer size upward to the next DP2 block size. For example, suppose you set the buffer at 30 bytes:

```
-SET BUFFERSIZE 30
```

Now if you enter the SHOW command, the buffer size is 512 bytes:

```
-SHOW
TYPE U
EXT ( 1 PAGES, 1 PAGES )
MAXEXTENTS 16
BUFFERSIZE 512
```

SHOW Command

Use the SHOW command to display the current settings of the file-creation attributes. Use the FUP SET and FUP RESET commands to set and reset these attributes.

The syntax of the SHOW command is:

```
SHOW [ / OUT <list-file> / ]
     [ <create-spec> [ , <create-spec> ]... ]
```

OUT <list-file>

names an existing disc file or a device to receive the listing output from the SHOW command.

<create-spec> is any one of these:

```
TYPE
CODE
EXT
REFRESH
AUDIT
REC
BLOCK
IBLOCK
COMPRESS
DCOMPRESS
ICOMPRESS
KEYLEN
KEYOFF
ALTKEY [ <key-specifier> ]
ALTKEYS
ALTFILE [ key-file-number ]
ALTFILES
ALTCREATE
PART [ partition-num ]
PARTS
PARTONLY
MAXEXTENTS
BUFFERSIZE
BUFFERED
```

→

```
AUDITCOMPRESS
VERIFIEDWRITES
SERIALWRITES
ODDUNSTR
```

Each <create-spec> you include in your SHOW command returns the current setting for that attribute. Omitting <create-spec> returns all current settings applicable to the current file type. (For more information about each <create-spec>, see <create-param> in the SET command.)

Considerations

- If you issue the SHOW command with a <create-spec> that is not currently set or one that is not applicable to the current value of TYPE, FUP returns only a prompt.
- Because the system rounds extent sizes upward if necessary, the SHOW display might list an EXT size that differs by one page from what is actually created. (For details, see the SET command EXT parameter.)
- You can set the extent size as the number of records in each extent. SHOW displays the extent size in that form. For example, if you enter:

  ```
  -SET EXT (100 RECS, 10)
  ```

  the SHOW display includes:

  ```
  EXT ( 100 RECS, 10 PAGES )
  ```

Example

If you enter FUP SHOW when the defaults are in effect for all file-creation attributes, the screen displays:

```
TYPE U
EXT ( 1 PAGES, 1 PAGES )
MAXEXTENTS 16
BUFFERSIZE 2048
```

SUBVOLS Command

Use the SUBVOLS command to display the names of all the subvolumes on a particular volume.

The syntax of the SUBVOLS command is:

```
SUBVOLS [ / OUT <list-file> / ]
        [ [\<system-name>.]$<volume-name> ]
```

OUT <list-file>

names a list file to receive the output of the SUBVOLS command.

\<system-name>.$<volume-name>

is the volume whose subvolume names are to be displayed. If you omit \<system-name>, FUP uses the current default system. If you do not include $<volume-name>, FUP lists the names of the subvolumes on the default volume.

Example

To display the names of all the subvolumes on the $SYSTEM volume, enter:

```
-SUBVOLS $SYSTEM
```

SYSTEM Command

Use the SYSTEM command to set the default system and, optionally, the default volume and subvolume names used by FUP in expanding file names. You can use the SYSTEM command only on a system that has a name, such as a system that is part of a network.

The syntax of the SYSTEM command is:

```
SYSTEM [ \<system-name>.[$<volume-name>][.<subvol-name> ] ]

\<system-name>

   names the new current default system.

$<volume-name>

   names the new current default volume.

<subvol-name>

   names the new current default subvolume.
```

Considerations

- To restore the default system name that was in effect when you started FUP, enter the SYSTEM command without parameters. The current default volume and subvolume names are not affected.
- FUP and COMINT keep separate defaults for system, volume, and subvolume. When you return to COMINT from FUP, the defaults are reset to the values that were in effect when you started FUP.

Examples

This example sets the \LONDON system as the current default for FUP:

```
-SYSTEM \LONDON
```

To set the current FUP defaults to the PAID subvolume on the $BILLS volume of system \NEWYORK, enter:

```
-SYSTEM \NEWYORK.$BILLS.PAID
```

VOLUME Command

Use the VOLUME command to change the current default volume and/or subvolume names used during the execution of FUP. The initial defaults are the subvolume and volume that were the current defaults when you started FUP. For more information, see the descriptions of the VOLUME command and the DEFAULT program in COMINT (Section 2).

The syntax of the VOLUME command is:

```
VOLUME  [\<system-name>.]{$<volume-name>.<subvolume-name>}
                         {$<volume-name>                 }
                         {<subvolume-name>               }
```

\<system-name>

sets FUP's current default system. You can set the default system only in a named system, such as one in a network.

$<volume-name>.<subvolume-name>

sets FUP's current default names for both the volume and subvolume.

$<volume-name>

sets FUP's current default volume name.

<subvolume-name>

sets FUP's current default subvolume name.

Considerations

- Entering VOLUME with no volume or subvolume name restores the defaults that were in effect when you started FUP.
- FUP and COMINT keep separate defaults for system, volume, and subvolume. When you return to COMINT from FUP, the defaults are reset to the values that were in effect when you started FUP.

Examples

This command sets FUP's current default subvolume to SUBVOL1 but does not change the current default volume or system:

```
-VOLUME SUBVOL1
```

This example sets FUP's current default volume to $BOOKS3 but does not change the current default subvolume or system:

```
-VOLUME $BOOKS3
```

To set FUP's current default system, volume, and subvolume to be \ITALY, $MILANO, and ARTWORK, enter:

```
-VOLUME \ITALY.$MILANO.ARTWORK
```

SECTION 4

THE BACKUP AND RESTORE, BACKUP2 AND RESTORE2 PROGRAMS

This section describes the syntax and operation of the BACKUP, BACKUP2, RESTORE and RESTORE2 programs. BACKUP and RESTORE work with files in the current disc-process format (DP1), while BACKUP2 and RESTORE2 can work with files in the new disc-process format (DP2), with DP1 files, or with both DP1 and DP2 files. If you are a new user of BACKUP and RESTORE, read the descriptions of these two utilities in the GUARDIAN Operating System User's Guide before using this section.

The command syntax and considerations for BACKUP are presented first in this section, followed by the syntax and considerations for BACKUP2. Similarly, the syntax and considerations for RESTORE precede those for RESTORE2.

USING THE BACKUP AND BACKUP2 PROGRAMS

The BACKUP and BACKUP2 programs copy files from disc to magnetic tape. Use BACKUP to back up DP1 files only. Use BACKUP2 to back up DP1 files, DP2 files, or both DP1 and DP2 files.

You can use backup tapes (with the RESTORE or RESTORE2 program) to transfer data from one system to another. Tape files act as "backups" that can replace lost data in the event of system catastrophes such as major power outages. If your disc files are damaged or lost, you can use the RESTORE or RESTORE2 program to restore the last backup tape you made. Together, BACKUP and RESTORE can be used to compress disc space, as described in the System Operator's Guide. BACKUP2 and RESTORE2 can compress both DP1 disc space and any DP2 disc space which contains files that are not audited by TMF.

The BACKUP program normally resides in the file named $SYSTEM.SYS<nn>.BACKUP, and the BACKUP2 program resides in the file $SYSTEM.SYS<nn>.BACKUP2. ($SYSTEM.SYS<nn> is the subvolume containing the GUARDIAN operating system image currently in use, where <nn> is a two-digit octal integer.)

You can enter BACKUP or BACKUP2 commands in either of these two ways:

- Through the GUARDIAN command interpreter (COMINT), by typing a complete BACKUP or BACKUP2 command at COMINT's colon prompt.

- Through the BACKUP or BACKUP2 program, by typing the name of the program at COMINT's colon prompt; this starts a BACKUP or BACKUP2 process, and the program displays its prompt, a question mark (?). Then you can enter the rest of your command at the question mark prompt. Type CTRL/Y to exit from BACKUP or BACKUP2.

With either of these methods, you can enter BACKUP or BACKUP2 command parameters through a disc file rather than interactively, as explained in the IN <file-name> description. Differences in the syntax and operation of BACKUP and BACKUP2 are summarized at the beginning of the description of BACKUP2 (see "Differences between the BACKUP and BACKUP2 Programs").

Mounting a BACKUP Tape

To mount a tape to use with BACKUP or RESTORE, follow these steps:

1. Wait until the program (BACKUP or RESTORE) displays this message:

   $<tape-device> NOT READY

   where $<tape-device> is the name of the tape drive specified in your BACKUP command. Respond by pressing RETURN.

2. The program immediately repeats the message:

   $<tape-device> NOT READY

   Respond by pressing RETURN.

3. Mount the tape. The program attempts the tape writing or reading operation at one-second intervals until the operation is successful and then continues.

To terminate the program when the TAPE NOT READY message appears, enter STOP or CTRL/Y.

To terminate the program at any other time, press the BREAK key. Then enter the STOP command at the command interpreter prompt.

The preceding steps also apply to all other tape-mounting messages from BACKUP or RESTORE, such as:

```
MOUNT NEXT REEL <n>

MOUNT PREVIOUS REEL <n> ?

MOUNT CORRECT REEL <n> ?

MOUNT NEXT TAPE SET ?

REEL <n> MOUNTED.  OK?
```

Mounting a BACKUP2 Tape

To mount a tape to use with BACKUP2 or RESTORE2, you follow the steps for mounting a tape to use with BACKUP or RESTORE. However, BACKUP2 and RESTORE2 display different prompts, as shown in this subsection.

To mount a tape to use with BACKUP2 or RESTORE2, follow these steps:

1. Wait until the program (BACKUP2 or RESTORE2) displays the message:

   ```
   $<tape-device> : Not ready?
   ```

   where $<tape-device> is the name of the tape drive you are using. Respond by pressing RETURN.

2. The program immediately repeats the message:

   ```
   $<tape-device> : Not ready?
   ```

   Respond by pressing RETURN.

3. Mount the tape. The program attempts the tape writing or reading operation at one-second intervals until the operation is successful and then continues.

To terminate the program when the "Not ready?" message appears, enter STOP or CTRL/Y.

To terminate the program at any other time, press the BREAK key. Then enter the STOP command at the command interpreter prompt.

The preceding steps also apply to all other tape-mounting messages from BACKUP2 or RESTORE2, such as:

Mount next reel #<n>?

Mount previous reel #<n>?

Mount next rebuild tape set?

NOTE

When you mount a tape, the program (BACKUP or BACKUP2, or RESTORE or RESTORE2) starts processing as soon as the command parameters are input and the tape device is ready. For this reason, you may not need to complete all the steps outlined here.

CAUTION

To prevent any writing to your tape during a restore operation, remove the write-enable ring when you mount each tape, or make sure that the tape you are using does not have a write-enable ring. Otherwise, any user who can access that tape drive can write to your tape at any time the drive is idle.

The BACKUP Program

The syntax of the command to start a BACKUP process is:

```
BACKUP [ / <run-option> [ , <run-option> ]... / ]
       [ <backup-parameters> ]
```

→

<run-option>

can be any <run-option> for the RUN command in COMINT. The two most common run options are:

IN <file-name>
OUT <list-file>

IN <file-name>

names a file from which BACKUP will take all command parameters (those parts of the BACKUP command that follow the slash at the end of the <run-option> list). If you omit the IN option, BACKUP uses the IN file that is in effect for the current COMINT process. (If you are using COMINT interactively, your home terminal is the IN file.) If you include the IN option, <file-name> must contain all the BACKUP command parameters that you want to include; you cannot enter additional parameters when you enter the command to start a BACKUP process. BACKUP ignores the IN file if your command contains any parameters other than the <run-option> list.

For <file-name>, you can specify the name of a disc file, of any readable device (such as a terminal), or of a process.

If you specify an interactive device as <file-name>, BACKUP displays its question mark prompt (?) on the device. Enter your BACKUP command parameters on this device in response to the "?" prompt. You can start new lines (by inserting an ampersand [&] and a RETURN) only where blanks occur in the syntax of the command. To indicate the end of the file, enter CTRL/Y. BACKUP then executes the command you entered.

To enter comment lines in an IN <file-name> for use with BACKUP, begin the line with two hyphens (--).

→

OUT <list-file>

specifies a file to receive all listing and error-message output. If you omit the OUT option, BACKUP sends output to the OUT file that is in effect for the current COMINT process. (If you are using COMINT interactively, COMINT's OUT file is your home terminal.)

If you specify a <list-file> that does not already exist, BACKUP creates an entry-sequenced disc file named <list-file> with a record length of 80 characters, block size of 2048 bytes, primary extent size of 4 pages, and secondary extent size of 16 pages.

See the description of the RUN command in COMINT for a complete <run-option> list (see Section 2).

<backup-parameters>

are the BACKUP command parameters and are defined as:

```
[ \<system-name>.]$<tape-file> ,  <fileset-list>

   [ , AUDITED                                    ]
   [ , BLOCKSIZE <data-record-size>               ]
   [ , DENSITY <density>                          ]
   [ , IGNORE                                     ]
   [ , LISTALL                                    ]
   [ , MSGONLOCK                                  ]
   [ , NOT <not-fileset-list>                     ]   ...
   [ , OLDFORMAT                                  ]
   [ , OPEN                                       ]
   [ , PARTIAL <partial-dump-date>                ]
   [ , PARTONLY                                   ]
   [ , START <fileset-name>                       ]
   [ , VERIFYTAPE                                 ]
   [ , VOL  {[ $<new-vol>.]<new-svol>}            ]
   [        {$<new-vol>               }           ]
```

→

If you do not include <backup-parameters>, BACKUP opens the IN <file-name> and reads it for the command parameters. (The home terminal is the default IN file for most interactive uses.) When your terminal is the IN file, BACKUP prompts you for parameters by displaying its question mark prompt (?).

[\<system-name>.]$<tape-file>

is the name of the magnetic tape unit to which the files are to be dumped.

\<system-name>

is the name of the system where $<tape-file> resides. If you omit \<system-name>, BACKUP assumes the system where you enter the BACKUP command.

$<tape-file>

is the name of the magnetic tape unit to be used. For $<tape-file>, you can specify either of these:

$<device-name>
$<logical-device-number>

- $<device-name> is the logical name of the magnetic tape unit, such as $TAPE1.
- $<logical-device-number> is the device number of the magnetic tape unit, such as $17.

<fileset-list>

designates one or more sets of related files to be backed up. For <fileset-list>, specify either of these:

<fileset>
(<fileset> [, <fileset>]...)

→

<fileset>

is a set of files you want to back up. For <fileset> you can specify any of these four:

```
[ $<volume>.][<subvol>.]<file-name>
[ $<volume>.][<subvol>.]*
[ $<volume>.]*.*
*.*.*
```

[$<volume>.][<subvol>.]<file-name>

is the name of a file you want to back up. If you omit $<volume>, BACKUP assumes the current default volume. If you omit <subvol>, BACKUP assumes the current default subvolume.

[$<volume>.][<subvol>.]*

indicates an entire subvolume. All the files in $<volume>.<subvol> are to be backed up. If you omit $<volume>, BACKUP uses the current default volume. If you omit <subvol>, BACKUP uses the current default subvolume.

[$<volume>.]*.*

indicates that all files in the volume $<volume> are to be backed up. If you omit $<volume>, BACKUP assumes the current default volume.

..*

means that all files in the system are to be backed up.

→

AUDITED

backs up files audited by the Transaction Monitoring Facility (TMF). If you omit AUDITED, audited files are not backed up.

BLOCKSIZE <data-record-size>

designates the size of the records written to the backup tape. Specify <data-record-size> as an even integer in the range between 2 and 30, inclusive, indicating the number of 1024-byte increments in each record. If you omit the BLOCKSIZE parameter, the value of 8 (8192 bytes) is assigned by default.

The actual size of the data written to tape is:

specified <data-record-size> in bytes + 4 bytes (for sequence number) + 2 bytes (for checksum)

CAUTION

The default block size for the NonStop 1+ system differs from the default block size for NonStop systems. For the NonStop 1+ system, the default is 2 (2048 bytes), and for NonStop systems, the default is 8 (8192 bytes). Because of this difference, you must specify BLOCKSIZE 2 when making a backup tape on a NonStop system that will later be restored on a NonStop 1+ system.

DENSITY <density>

indicates the tape recording density (for the Model 5106 tape drive only), as follows:

<density>	Resulting Recording Density (in bpi)
GCR or 6250	6250
PE or 1600	1600
NRZI or 800	800

→

IGNORE

means that BACKUP ignores certain data errors on the disc. If possible, BACKUP writes the invalid data to tape; otherwise zeros are written. Ignoring of errors is done on a sector-by-sector basis (a sector is 512 bytes); an error message is issued to the list file for each bad sector. If you omit IGNORE and a data error occurs, BACKUP backspaces over any data written for the current file and proceeds with the next file.

LISTALL

lists the names of all files dumped, as well as the names of files that cause errors. If you omit LISTALL, BACKUP lists only those file names associated with error messages. BACKUP sends the LISTALL listing to the OUT <list-file> (if you specify one) or to the device where you entered the BACKUP command.

MSGONLOCK

displays the following message at your terminal if a file or record you want to back up is locked:

```
FILE IS LOCKED, FILENAME: <file-name>
ENTER SKIP TO SKIP THE FILE OR CR TO WAIT
```

To skip the locked file and continue immediately with the backup, enter SKIP; any records remaining on the tape are back-erased. (If the first file on the reel is locked and you type SKIP, a Beginning-of-Tape (BOT) condition occurs, and the BACKUP process aborts; this does not happen, however, with BACKUP2.)

To back up the locked file, press RETURN. BACKUP waits for the file to become unlocked and then copies the file to tape.

→

If you do not specify the MSGONLOCK parameter, BACKUP waits for a locked file but does not display a message.

NOT <not-fileset-list>

is a list of files that are not to be included in the backup. The form of <not-fileset-list> is the same as for <fileset-list>.

OLDFORMAT

creates an old backup-format tape. You can restore files from old format tapes by either the RESTORE program that was available prior to the Tandem software release for April 1, 1982, or the RESTORE program available at and after that release.

NOTE

You cannot use the VERIFYTAPE, DENSITY, and BLOCKSIZE parameters in the same BACKUP command with OLDFORMAT. (Checksum values are not generated with OLDFORMAT.)

OPEN

backs up files even if they are currently open with write or read/write access (unless the files are also open with exclusive access). If you do not include OPEN, BACKUP does not back up files that are currently open with write or read/write access.

PARTIAL <partial-dump-date>

backs up only files modified since <partial-dump-date>. However, disc file header entries for other files are dumped so that the latest directory can be reconstructed when a RESTORE is done from this and previous backup tapes (the previous tapes contain files not dumped during this operation). For <partial-dump-date>, specify either of these:

```
<day> <month-name>  <year> , <hour>:<minute>
<month-name> <day>  <year> , <hour>:<minute>
```

For <day>, give the date in the month, such as 1, 2, 15, 31.

For <month-name>, give the first three letters of the name of the month, such as JAN, FEB, MAR, DEC.

For <year>, give four digits, such as 1976, 1985.

For <hour>, give a one- or two-digit number, such as 0 for midnight, 1 for 1 a.m., 22 for 10 p.m.

For <minute> give two digits, such as 00 for the hour, 20 for 20 minutes past the hour, and 55 for five minutes before the hour.

PARTONLY

backs up any nonpartitioned files and primary or secondary partitions defined in the <fileset-list>, but does not back up entire partitioned files. Use PARTONLY when you want to back up only the files and partitions of files residing on a particular volume.

If you omit PARTONLY and a primary partition is defined in the <fileset-list>, the entire partitioned file (all partitions) is backed up. Secondary partitions defined in <fileset-list> are not backed up.

→

START <fileset-name>

indicates where in the <fileset-list> BACKUP is to begin. (Files are sorted according to the ASCII collating sequence.) Specify <fileset-name> as a fully qualified file name indicating the file, subvolume, or volume where you want BACKUP to begin copying files. <fileset-name> must be a subset of <fileset-list> and can have an asterisk (*) as the subvolume or file name, but not as the volume name.

The following nine cases are the only valid ways of using the START parameter with BACKUP:

<fileset-list>	<fileset-name>	Action Taken
..*	$VOL.*.*	Start with the first file on the first subvolume on volume $VOL and continue through all files on all subvolumes and volumes following $VOL.
..*	$VOL.SVOL.*	Start with the first file on $VOL.SVOL and continue through all files in subvolume SVOL and all subvolumes following SVOL, and all volumes following $VOL.
..*	$VOL.SVOL.FILE	Same as the previous case except start with the file named FILE on $VOL.SVOL.
$VOL.*.*	$VOL.*.*	Start with the first file of the first subvolume on the $VOL volume and continue through all files in all subvolumes on $VOL only.

→

<fileset-list>	<fileset-name>	Action Taken
$VOL.*.*	$VOL.SVOL.*	Start with the first file on $VOL.SVOL and continue through all files on SVOL following $VOL.SVOL.
$VOL.*.*	$VOL.SVOL.FILE	Same as the previous case except start with the file named FILE on $VOL.SVOL.
$VOL.SVOL.*	$VOL.SVOL.*	Start with the first file on $VOL.SVOL and continue through all files on $VOL.SVOL.
$VOL.SVOL.*	$VOL.SVOL.FILE	Same as the previous case except start with the file named FILE on $VOL.SVOL.
$VOL.SVOL.FILE	$VOL.SVOL.FILE	Start and end with the file named FILE on $VOL.SVOL.

VERIFYTAPE

verifies the volume label, file label, and data records written to the backup tape. A checksum is appended to each of these items when they are first written to tape (provided that you do not include the OLDFORMAT option, discussed later in this section).

If you include VERIFYTAPE, BACKUP repositions the tape to the beginning of the last file dumped, and reads each record (including the file label) after it reaches the end of each file or the end of the tape (whichever occurs first). As it reads each record, the BACKUP program regenerates a checksum for that record and compares the checksum to the one actually

→

read back. The file-label checksum for each file is compared to that in the saved file label generated when the checksum was written.

VOL { [$<new-vol>.]<new-svol> }
    { $<new-vol>   }

gives a new volume or subvolume name (or both) for the tape versions of the files being backed up. If you omit the VOL option, BACKUP does not change the names of the files that are backed up. If you include $<new-vol>, BACKUP gives the files in <fileset-list> the tape volume name $<new-vol> in place of their original disc volume name. If you include <new-svol>, files in <fileset-list> receive the tape subvolume name <new-svol> in place of their original disc subvolume name.

You can specify a new volume name with the VOL option only if all the files in <fileset-list> reside in the same volume. Similarly, you can specify a new subvolume name with the VOL option only if all files in <fileset-list> reside in the same subvolume.

Listing Format

When you perform a backup operation with the BACKUP program, BACKUP first displays its one-line program banner at the OUT <list-file>. Then it echoes the IN <file-name> (if you included one in your BACKUP command) and displays its program listing as shown in Figure 4-1. If you include the LISTALL parameter, the BACKUP listing includes all files specified in the BACKUP command. If you omit LISTALL, BACKUP lists only those file names for which a comment (error or warning message) is listed.

```
VOLNAME SVOLNAME FILENAME REEL ERROR ADDRESS<dumpdate><vsn>[PART]
   .       .        .     .                    <comment>
   .       .        .     .                       .
   .       .        .     .                       .

FILES DUMPED=<num>   FILES NOT DUMPED=<num>   [DATA ERRORS=<num>]
```

Figure 4-1. BACKUP Program Listing Format

The listing headers and variables that appear in Figure 4-1 are:

VOLNAME SVOLNAME FILENAME	heads a list of the file names specified to the BACKUP program. The file name is indented one space if the file is a secondary partition or if it is a primary partition and you included PARTONLY.
REEL	indicates the magnetic tape reel where the file was dumped. The first reel used in a backup operation is reel one. Files dumped across multiple reels are listed once for each reel.
ERROR	is the file-system error number of any error preventing the normal backup of a file.
ADDRESS	is the byte address of a sector in error relative to file byte zero.
<dumpdate>	is the date and time of day when the backup started.

<vsn>	is the version level of the operating system at the time the backup operation was performed.
PART	means that the PARTONLY option was specified in the BACKUP command.
<comment>	is a message that might be listed on the OUT <list-file> for individual files. Possible comments include:
FILE OPEN	is a warning message. The OPEN option was specified, and the listed file is open.
DATA BAD	is a warning message. You included the IGNORE option, and the sector indicated by ADDRESS contains invalid data. The invalid data (that is, 512 bytes) was written to tape.
DATA LOST	is a warning message. The IGNORE option was specified, and the sector indicated by ADDRESS contains invalid data that could not be read. Zeros are written for the sector to the tape file.
NOT DUMPED	indicates that this file was not dumped because of the file-system error listed under ERROR in the listing file.
SET ABORTED	indicates that the rest of the file set associated with this file was not dumped because of the file-system error listed under ERROR in the listing file.
FILES DUMPED=<num>	is the number of files written on tape.
FILES NOT DUMPED=<num>	indicates the number of files that were specified for backup but were not dumped.
DATA ERRORS=<num>	appears only when you include the IGNORE option. <num> is the number of sectors that could not be read successfully.

Considerations

- Entering parameters in an IN <file-name> allows you to specify a larger <fileset-list> than can fit on a single COMINT command line. You can specify over 100 file sets in an IN file. These filesets can be a part of either <fileset-list> or <not-fileset-list>. By contrast, only about 50 file sets can fit in a single COMINT command that extends over more than one line.

- Use the START parameter with BACKUP in these two situations only: when all the files in the <fileset-list> reside on a single volume; or when all the files in <fileset-name> reside on the first volume specified in <fileset-list>.

- If the primary partition of a partitioned file is included in <fileset-list> and you do not include the PARTONLY option, all partitions are automatically backed up. If an extra partition (that is, a partition numbered between 1 and 15) is included in the <fileset-list>, it is not backed up unless the primary partition is also in the <fileset-list>.

- In backing up files, BACKUP attempts to open files with protected access. This means that processes other than BACKUP can read the file, but they cannot write to it. If you include the OPEN option, and BACKUP encounters a file that is open, and an attempt to open the file with protected access fails, BACKUP then attempts to open the file with shared access. See the ENSCRIBE Programming Manual for a discussion of file-access modes.

- Files that are open with exclusive access cannot be backed up.

- Always close files if possible before backing them up. You can, however, back up files that are open for read-write access by including the OPEN option in the BACKUP command. You should be very cautious about backing up any open files, especially key-sequenced files. If a file is modified while it is backed up and the file is later restored, file-system error 59 (FILE IS BAD) can occur when the file is read, and you may lose your data.

  If you want to back up files with the OPEN option, you should:

  --Issue a Peripheral Utility Program (PUP) REFRESH command before backing up files (this command updates file labels on disc)

  --Be certain the files being backed up are NOT written to during the backup

- If you enter the DENSITY parameter for a tape drive other than a Model 5106, this message appears on your terminal and the BACKUP process terminates:

  DRIVE DOES NOT SUPPORT DENSITY SELECTION

- BACKUP checks the disc-process type of all files in the <fileset-list> you specify. BACKUP then skips file sets that do not reside on a DP1 volume and displays this message in the comment column of the BACKUP listing:

  NOT DP1 VOLUME

  If you get this message, use the BACKUP2 program to back up the DP2 files.

Considerations for Files Audited by TMF

- To back up files that are audited by TMF, you must include the AUDITED option. TMF has its own recovery mechanisms for audited files, and you should not use BACKUP and RESTORE in place of them, except in extraordinary circumstances. You might, however, want to use BACKUP and RESTORE:

  --For compressing data in a disc volume containing audited files (See your System Operator's Guide for information about disc compression.)

  --For transporting audited files to another system

  --During a TMF emergency

  --For archiving files when the RECOVERY OFF option of TMF is used

  If you do not include the AUDITED option, and BACKUP encounters an audited file, BACKUP skips the file and sends the message *** AUDITED FILE SKIPPED to the output file.

- No entries are made in the TMF catalog for audited files that are backed up.

- The state of TMF has no effect on the BACKUP program.

Checksum Errors

If you enter the VERIFYTAPE parameter and a checksum error is detected, the following messages appear at your terminal and on the device designated as the output file:

For a beginning volume label:

```
DETECTION OF CHECKSUM ERROR ON TAPE READ
VOLUME LABEL CHECKSUM ERROR, LENGTH: <volume-label-length>
```

For an ending volume label:

```
DETECTION OF CHECKSUM ERROR ON TAPE READ
ENDING VOLUME LABEL CHECKSUM ERROR,LENGTH:<volume-label-length>
```

For a file label:

```
DETECTION OF CHECKSUM ERROR ON TAPE READ
FILENAME: <file-name>
FILE LABEL CHECKSUM ERROR, LENGTH: <file-label-length>
```

For a data record:

```
DETECTION OF CHECKSUM ERROR ON TAPE READ
FILENAME: <file-name>
BYTEOFFSET: <byte-offset>
```

If you do not specify the IGNORE parameter and a checksum error occurs, the following prompt is displayed at your terminal:

```
ENTER CR TO CONTINUE, ENTER STOP TO ABORT
```

To continue the backup process, press RETURN. To abort the backup process, enter STOP.

Examples

The following command backs up to the tape mounted on the device $TAPE all the files in the subvol MYSVOL on the volume $MYVOL. The LISTALL option causes all the file names to be listed on the terminal from which you started the BACKUP process:

```
:BACKUP $TAPE, $MYVOL.MYSVOL.*, LISTALL
```

This pair of commands backs up the file $DATA.RECRDS.MYFILE, all files in SVOL2 on the default volume (that is, $DATA), and all files in the default subvolume (RECRDS) on $VOL2:

```
:VOLUME $DATA.RECRDS
:BACKUP $TAPE, (MYFILE, SVOL2.*, $VOL2.*)
```

These commands back up all files on the $SYSTEM volume and list the file names on the home terminal:

```
:VOLUME $SYSTEM.SYSTEM
:BACKUP $TAPE, *.*, LISTALL
```

These commands back up all files in the subvolume ACCTS on the $DATA volume. The associated file names are listed on the device $LP (a line printer):

```
:VOLUME $DATA.ACCTS
:BACKUP /OUT $LP/ $TAPE, *, LISTALL
```

To back up all files in the current default subvolume on volume $STORE1 except the file ACCT5, enter:

```
:VOLUME $STORE1
:BACKUP $TAPE, *, NOT ACCT5
```

This command backs up all files in the system except those on the volume $SYSTEM. The listing is sent to the home terminal:

```
:BACKUP $TAPE, *.*.*, NOT $SYSTEM.*.*, LISTALL
```

This command backs up all files on the volume $LIB1. Any primary or secondary partitions on $LIB1 are backed up; partitions on any other volume are not backed up. The BACKUP listing is displayed at the home terminal:

```
:BACKUP $TAPE, $LIB1.*.*, PARTONLY, LISTALL
```

The next example shows a backup operation that encounters errors in three files. Because IGNORE was not specified, the two files containing data errors are not dumped. Because OPEN was not specified, the open GPLIB file is not dumped. Because LISTALL was omitted, only files associated with comments are listed:

```
   :BACKUP $TAPE, *

VOLNAME SVOLNAME FILENAME REEL ERROR  ADDRESS 14 MAY 85 09:05 B00

$SYSTEM  SYSTEM  COPY      01   120              *** NOT DUMPED
$SYSTEM  SYSTEM  GPLIB     01   012              *** NOT DUMPED
$SYSTEM  SYSTEM  PRIVDECS  01   120              *** NOT DUMPED

FILES DUMPED=00014   FILES NOT DUMPED=00003
```

This final example shows what happens when you specify IGNORE and an error occurs, and when you specify OPEN and an open file is dumped. The file COPY is readable but contains an error in the sector starting at relative byte 512. The sector containing bad data is written to tape. The file GPLIB is open; therefore the "*** FILE OPEN" message is a warning. The file PRIVDECS has two sectors containing errors that prevent any data (even bad data) from being read; zeros are written to tape for the sectors in error:

```
   :BACKUP $TAPE,*,IGNORE,OPEN

VOLNAME SVOLNAME FILENAME REEL ERROR  ADDRESS 14 MAY 85 09:10 B00
$SYSTEM  SYSTEM  COPY      01   120  0000000512  *** DATA BAD
$SYSTEM  SYSTEM  GPLIB     01                   *** FILE OPEN
$SYSTEM  SYSTEM  PRIVDECS  01   120  0000001536  *** DATA LOST
$SYSTEM  SYSTEM  PRIVDECS  01   120  0000002048  *** DATA LOST

FILES DUMPED=00016   FILES NOT DUMPED=00000   DATA ERRORS=00003
```

Differences between the BACKUP and BACKUP2 Programs

The BACKUP program can back up only DP1 files. BACKUP2 can back up DP1 files, DP2 files, or a mix of DP1 and DP2 files. The basic operation of BACKUP2 is similar to that of BACKUP, but BACKUP2 has expanded capabilities and a different listing format (similar to that of the FUP INFO command).

BACKUP2 has these additional command parameters:

- DP1FORMAT causes DP2 disc files to be converted to DP1 tape files.
- DP2FORMAT causes DP1 disc files to be converted to DP2 tape files.
- OLDFORMAT causes both DP1 and DP2 files to be written as old backup-format tape files; DP2 files also go through an intermediate conversion to DP1 disc files.
- EXT specifies the primary and secondary extent sizes for the destination file in a file conversion (when converting files from one disc-process type to another).
- DSLACK sets the minimum percentage of slack space in data blocks in a file conversion.
- ISLACK sets the minimum percentage of slack space in index blocks in a file conversion.
- VERIFYREEL causes BACKUP2 to verify each reel after writing it.
- PART renames the secondary partitions of a partitioned file.
- ALTFILE renames alternate-key files.

BACKUP can create tapes in either of these two formats: (1) the default BACKUP format or DP1, and (2) the old-backup format, created when you include the OLDFORMAT option. BACKUP2 can create tapes in any of three formats: (1) old-backup format or tape version 0, (2) DP1 format or tape version 1, and (3) DP2 format or tape version 2.

With BACKUP2, the tape format is determined by the format of the files you are backing up and by the options you include in your BACKUP2 command. If you are backing up only DP1 files, BACKUP2 creates a tape in version 1; however, you can produce tape version 0 by including OLDFORMAT, or tape version 2 by including DP2FORMAT. If you are backing up any DP2 files, the tape is in

version 2 unless you specify DP1FORMAT for tape version 1, or OLDFORMAT for tape version 0. See the syntax and listing-header descriptions of BACKUP2 for more information.

The BACKUP2 Program

This syntax description includes only the parameters that are unique to BACKUP2. Parameters that are the same for BACKUP and BACKUP2 are described in the syntax for BACKUP, as indicated by the note "See the syntax entry for BACKUP."

The syntax of the command to start a BACKUP2 process is:

```
BACKUP2 [ / <run-option> [ , <run-option> ]... / ]
        [ <backup2-parameters> ]

<run-option> can be any of those for the RUN command in
COMINT.  Two common <run-option>s are:

   IN <file-name>

      See the syntax entry for BACKUP.

   OUT <list-file>

      See the syntax entry for BACKUP.
```

<backup2-parameters>

are the BACKUP2 command parameters, and are defined as:

```
[ \<system-name>.]$<tape-file> ,  <fileset-list>

[ , ALTFILE ( <key-file-num> ,
            [ $<vol>.][<subvol>.]<file-name> )
  , AUDITED
  , BLOCKSIZE <data-record-size>
  , DENSITY <density>
  , DP1FORMAT
  , DP2FORMAT
  , DSLACK <percentage>
  , EXT  { <extent-size>                                  }
         { ( <pri-extent-size> , <sec-extent-size> )      }
  , IGNORE
  , ISLACK <percentage>
  , LISTALL
  , MSGONLOCK                                                ...
  , NOT <not-fileset-list>
  , OLDFORMAT
  , OPEN
  , PART ( <secondary-partition-num>
       , [ [ \<system-name>.]$<volume-name> ]
         [ , <pri-extent-size> , [ <sec-extent-size> ]])
  , PARTIAL <partial-dump-date>
  , PARTONLY
  , START <fileset-name>
  , VERIFYREEL
  , VERIFYTAPE
  , VOL  { [ $<new-vol>.]<new-svol> }
         { $<new-vol>               }
]
```

If you do not include any <backup2-parameters>, BACKUP2 opens the IN <file-name> and reads it for the command parameters. If you do not specify an IN option, BACKUP2 uses the IN file in effect for the current COMINT (the home terminal is the IN file for most interactive uses of COMINT). When the terminal is serving as an IN file, BACKUP2 prompts you for parameters by displaying its prompt, a question mark (?).

→

[\<system-name>.]$<tape-file>

See the syntax entry for BACKUP.

<fileset-list>

See the syntax entry for BACKUP.

ALTFILE (<key-file-number> ,
[$<vol>.][<subvol>.]<file-name>)

changes the name for an alternate-key file in the file label of the primary-key file. Specify <key-file-number> as an integer in the range from 0 to 255, inclusive, designating the alternate-key file you are naming. For <file-name>, give the new name of the alternate-key file. You can include new volume and subvolume names for the alternate-key file by including $<vol> and <subvol>.

AUDITED

See the syntax entry for BACKUP.

BLOCKSIZE <data-record-size>

See the syntax entry for BACKUP.

DENSITY <density>

See the syntax entry for BACKUP.

→

DP1FORMAT

causes all files to be written to tape as DP1 files (that is, tape version 1). Any DP2 files encountered are first converted on disc into temporary DP1 files and then written to tape in DP1 format. You can omit this option if your <fileset-list> contains only DP1 files, and you want these files to remain in DP1 format.

DP2FORMAT

causes all files to be written to tape in the format of DP2 files (that is, tape version 2). Any DP1 files encountered are first converted on disc into temporary DP2 files and then written to tape. You can omit DP2FORMAT if the <fileset-list> contains at least one DP2 file, and you want your backup tape in DP2 format.

DSLACK <percentage>

used only when converting key-sequenced files from DP1 format to DP2 format; sets the minimum percentage of slack space in data blocks (see the LOAD command in FUP). The default DSLACK percentage is 0.

```
EXT  { <extent-size>                                  }
     { ( <pri-extent-size> , <sec-extent-size> )      }
```

used only when converting files from DP1 format to DP2 format; sets the extent size of the file copy to be made on tape. For <extent-size>, <pri-extent-size>, and <sec-extent-size>, you can specify an integer in the range from 1 to 65,535, inclusive, designating the extent size in pages (units of 2048 bytes).

→

IGNORE

works for BACKUP2 as it does for BACKUP. For DP2 files, however, BACKUP2 ignores errors on a block-by-block basis. The length of this block for structured files is the data-block length for the file; for unstructured files, this block is the file's buffer length. The default block size for structured files is 1024 bytes; the default buffer size for unstructured files is 4096 bytes. For more information, see the FUP SET command in Section 3.

ISLACK <percentage>

used only when converting key-sequenced files from DP1 format to DP2 format; sets the minimum percentage of slack space in index blocks (see the LOAD command in FUP). The default ISLACK percentage is 0.

LISTALL

See the syntax entry for BACKUP.

MSGONLOCK

displays messages as does MSGONLOCK with BACKUP, but does not abort if the first file is locked and you specify SKIP. The message is:

```
File or record lock encountered in file: <file-name>
Enter SKIP to skip the file or carriage return to
  wait for unlock?
```

Because bulk I-O is used for DP2 files, record locks will not cause this message to be displayed when you are backing up DP2 files.

NOT <not-fileset-list>

See the syntax entry for BACKUP.

→

OLDFORMAT

is the same for BACKUP2 as for BACKUP, except that BACKUP2 with OLDFORMAT converts DP2 files into temporary DP1 disc files before writing them to tape in the old backup-format.

NOTE

You cannot include VERIFYREEL, VERIFYTAPE, or BLOCKSIZE in the same BACKUP2 command with OLDFORMAT. (Checksum values are not generated with OLDFORMAT.)

OPEN

See the syntax entry for BACKUP.

PART (<secondary-partition-num>
, [[\<system-name>.]$<volume-name>]
[, <pri-extent-size> , [<sec-extent-size>]])

specifies a new name for a secondary partition of a partitioned file. The PART option causes the secondary-partition name to be inserted in the file label of the primary partition. The PART parameters <secondary-partition-num>, \<system-name>, and <volume-name> are described in the SET command in Section 3.

When you specify PART, the original name of the partition indicated by <secondary-partition-num> is replaced by the new partition name you specify. The <secondary-partition-num> should already exist in the source file.

If you omit the volume name, BACKUP2 uses the original volume name of the secondary partition.

→

You can also optionally specify the destination partition extent sizes. Specify <pri-extent-size> and <sec-extent-size> as integers in the range from 1 to 65,535, inclusive, indicating the size of the primary and secondary extents, respectively, in units of pages (1 page is 1024 bytes). BACKUP2 uses the extent sizes only when converting files from one disc-process type to another.

If you include both PART and PARTONLY, BACKUP2 changes the name of the secondary partition in the file label of the primary partition. If you include PART without PARTONLY, BACKUP2 changes the secondary-partition name in the primary-partition file label and changes the actual file name of the secondary partition as well.

PARTIAL

See the syntax entry for BACKUP.

PARTONLY

is the same for BACKUP2 as for BACKUP, except that this option causes BACKUP2 to skip certain individual partitions when converting files from one disc-process type to another (see the File Conversion Manual for details). An error message gives the names of the skipped files.

START <fileset-name>

See the syntax entry for BACKUP.

VERIFYREEL

verifies each reel after it is written. BACKUP2 rewinds and rereads the tape, then rewinds and unloads the tape. Verification involves reading each record to make sure it is readable and checking that the tape checksum is correct. Tape-record sequence numbers are also checked.

VERIFYTAPE

does the same checking as for VERIFYTAPE with BACKUP but also checks sequence numbers of each tape record.

VOL { [$<new-vol>.]<new-svol> }
    { $<new-vol>   }

See the syntax entry for BACKUP.

Listing Format

Figure 4-2 shows the listing format used by the BACKUP2 program. When you run a BACKUP2 process, the program first displays a one-line program banner at the OUT file (this is usually your terminal). Then it echoes the IN file (if you included one in your command) and displays its listing as shown in Figure 4-2.

If you include the LISTALL parameter, BACKUP2 gives a listing for all files that you specify to be backed up. If you omit LISTALL, the BACKUP2 listing includes only those files for which a comment (either an error message or warning) is also listed.

```
[*WARNING*   This tape can only be restored with TNS/II RESTORE2.]
Tape: $<tape>  Operating System: <vsn>  Tape Version: <v>
Backup options: <params>
[Partial Time: <date> <time> ]
Backup time:  <date> <time>                                    Page:<n>

Reel: <n>      Code     EOF   Last modif  Owner  RWEP  Type   Rec Block

$<vol>.<subvol>
 <filename>    <code>  <eof> <date><time>  <g,u> <rwep><type> <rec><bl>
    .    .       .       .     .       .      .     .
    .    .       .       .     .       .      .     .
[            *ERROR*    <comment>          ]
[            *WARNING*  <comment>          ]

Files dumped = <n>  Files not dumped = <n>  [ Data errors = <n> ]
```

Figure 4-2. BACKUP2 Program Listing Format

The listing headers and variables that appear in Figure 4-2 are:

WARNING This tape can only be restored with TNS/II RESTORE2.

appears only when you are generating a tape of version 2 (DP2 format) and warns you that you must use RESTORE2 to restore these files to disc.

TAPE: $<tape> is the name of the tape drive where the listed files were written.

Operating System:<vsn> is the version level of the operating system at the time the backup operation was performed.

Tape Version: <v> identifies the version of the tape format. <v> can be 0 (for old format), 1 (for DP1 format), or 2 (for DP2 format).

Backup options: <params> lists the command parameters that might affect the restoration of these files. The tape format determines the options that are listed. That is, if the tape is in the old format, only one option is listed ([NO] PARTONLY). For a DP1 format tape, one additional option appears (BLOCKSIZE <n>). For a DP2 format tape, three more options appear ([NO] AUDITED, [NO] OPEN, and [NO] IGNORE). For a discussion of the significance of these options, see the "Considerations" for RESTORE2.

Partial Time:<date><time> is the date and time specified with the PARTIAL option in your BACKUP2 command; appears only for tape version 2 (DP2 format).

Backup time:<date><time> gives the date and time that the backup operation started.

Page: <n> is the page number of the listing.

Reel: <n> indicates the magnetic tape reel where the file was dumped. The first reel used in a backup operation is reel one. Files dumped across multiple reels are listed once for each reel.

$<vol>.<subvol>	is the volume and subvolume name of the files that follow in the listing.
<file-name>	is the file name of a file in the file set to be backed up.
CODE <code>	is the file code of the listed file. Reserved file codes are listed in Table 3-1 in the FUP section of this manual.
EOF <eof>	is the end-of-file value (the size of the file in bytes).
Last modif <date><time>	is the date and time that the listed file was last modified.
Owner <g,u>	is the group ID and user ID of the owner of the listed file.
RWEP <rwep>	is the security of the listed file.
Type <type>	is the file type of the listed file. <type> can be any of these: <null> = unstructured file R = relative file E = entry-sequenced file K = key-sequenced file <t>A = file with alternate key P<t> = partitioned file XP<t> = file is an extra partition (secondary partition) where <t> can be one or more of R, E, K, P, or X.
Rec <rec>	for ENSCRIBE structured files, is the file's logical record length in bytes. This field is blank for unstructured files.
Block <bl>	for ENSCRIBE structured files, is the file's block length in bytes. This field is blank for unstructured files.
ERROR <comment>	is any file-system error that prevents the normal backup of a file.
WARNING <comment>	is a warning about the file being backed up.

Files dumped = <n>	is the number of files written on tape.
Files not dumped = <n>	is the number of files that were specified for backup but were not written to tape.
Data errors = <n>	is the number of data errors encountered in the files that were backed up.

Considerations

- When you use BACKUP2 to convert key-sequenced files from DP1 to DP2 or vice versa, you can specify the data and index slack space with the DSLACK and ISLACK parameters, just as you do with the LOAD command in FUP. Note, however, that the index-block size for DP2 files is the same as the data-block size.

- You can use BACKUP2 to copy an entry-sequenced DP1 file to a DP2-format tape. However, if the DP1 file has a non-DP2 block size (1536, 2560, 3072, or 3584 bytes), this operation changes record addresses in the file. If BACKUP2 converts a primary DP1 entry-sequenced file to DP2 format, a message is displayed warning you that you must perform a FUP LOADALTFILE operation on all the alternate-key files of this primary file.

- BACKUP2 will back up entire partitioned files that have some secondary partitions on remote systems if you include the DP2FORMAT option.

- Both BACKUP2 and RESTORE2 try to adjust extent sizes of individual partitions when converting partitioned files from one disc process to another. If all the records of relative and entry-sequenced files can remain in the same partition, then the file is converted; otherwise, it is skipped.

- When BACKUP2 converts files from one format to another, it creates temporary disc files on the current default subvolume. The temporary files, which are in the destination file format, have names that begin with ZZCV. If you terminate BACKUP2 early, the default subvolume may contain temporary files. To save disc space, purge any remaining ZZCV files.

Examples

The first example demonstrates converting several types of files from DP1 to DP2 using BACKUP2. All the files in the subvolume $DATA.RECDS are backed up to $TAPE4. The backup tape will be in tape version 2 because the DP2FORMAT option was included. The tape files will get the volume name specified with the VOL option (that is, $DP2.RECDS):

```
:BACKUP2 $TAPE4, $DATA.RECDS.*, LISTALL, DP2FORMAT,&
:VOL $DP2.RECDS
```

```
Guardian File BACKUP2 Program - T9075B00 - (18MAR85)   System: \TS
*WARNING*  This tape can only be restored with TNS/II RESTORE2.
Tape: $TAPE4  Operating System: B00  Tape Version: 2
Backup options: NO AUDITED, BLOCKSIZE 8, NO IGNORE, NO OPEN, NO PARTONLY
Backup time:  9Nov84 17:47                                          Page: 1

Reel: 1        Code          EOF     Last modif    Owner RWEP   Type    Rec Block

$DP2.RECDS
 ENTALT                    62464  9Nov84 14:37   1,34  CUCU    E       80  1024
 ENTALT1                   82944  9Nov84 14:37   1,34  CUCU    K       27  10

$DP2.RECDS
 ENTPART                   32768  2Nov84 14:40   3,16  CUNU   PE       80  1024
$SYSTEM.RECDS
 ENTPART                   29696  2Nov84 14:40   3,16  CUNU  XPE       80  1024
$VOL3.RECDS
 ENTPART                       0  2Nov84 14:40   3,16  CUNU  XPE       80  1024

$DP2.RECDS
 KEYALT                    62464  9Nov84 14:37   1,34  CUCU    E       80  1024
 KEYALT1                   72704  9Nov84 14:38   1,34  CUCU    K       19  1024
 KEYALT2                   13312  9Nov84 14:38   1,34  CUCU    K       27  1024

$DP2.RECDS
 KEYPART       *ERROR*  $DATA.RECDS.ZZCV4032:  File aborted (Error 45).
 KEYPART       *ERROR*  Entire partitioned file aborted.

$DP2.RECDS
 RELALT                   279552  9Nov84  14:38  1,34  CUCU    R       80  1024
 RELALT1                   72704  9Nov84  14:38  1,34  CUCU    K       19  1024
 RELALT2                   13312  9Nov84  14:38  1,34  CUCU    K       27  1024

$DP2.RECDS
 RELPART                   63488  3Nov84  16:17  3,16  CUNU   PR       80  1024
$VOL3.RECDS
 RELPART                    8192  3Nov84  16:17  3,16  CUNU  XPR       80  1024
$SYSTEM.RECDS
 RELPART                       0  3Nov84  16:17  3,16  CUNU  XPR       80  1024

Files dumped = 10  Files not dumped = 1
```

The files specified to be backed up in this example are different types of structured files:

- ENTALT is an entry-sequenced file with alternate-key files.
- ENTPART is an entry-sequenced, partitioned file (that is, an entry-sequenced file that has secondary partitions on other volumes).
- KEYALT is a key-sequenced file with alternate-key files.
- KEYPART is a key-sequenced, partitioned file.
- RELALT is a relative file with alternate-key files.
- RELPART is a relative, partitioned file.

Note that when you use BACKUP2 to back up a partitioned file, such as ENTPART, the volume names of the secondary partitions do not change (even if you include the VOL option to rename the primary partition). To restore a partitioned file onto a system whose volume names differ from the volume names of the secondary partitions, you must use RESTORE2 with the PART option to rename the secondary partitions (see the examples for RESTORE2).

In this example, an error occurs with KEYPART (Error 45) because the size of the temporary disc file used by BACKUP2 for converting files from DP1 to DP2 is too small for this key-sequenced file. (To back up this file, include the EXT parameter and specify larger extent sizes.)

The next example creates a backup tape in DP1 format (tape version 1), copies files from the subvolume $VOL1.FILES, gives the files the volume and subvolume names $SPOOL.RECDS, and sends a complete listing to the file BUPFILE in the current default subvolume:

```
:BACKUP2 /OUT BUPFILE/ $TAPE3, $VOL1.FILES, LISTALL, &
:DP1FORMAT, VOL $SPOOL.RECDS
```

USING THE RESTORE AND RESTORE2 PROGRAMS

Use the RESTORE and RESTORE2 programs for either of these purposes:

- To restore files (that is, to copy files from magnetic tape onto disc)
- To list the contents of a tape without restoring data

The two functions of these utilities (restoring and displaying) are presented separately in the following syntax descriptions. Because many users want to list the contents of a tape before restoring files, the display forms of RESTORE and RESTORE2 are described first, before the restore forms.

Although RESTORE2 has many similarities to RESTORE, there are significant differences in function and operation. Differences between the two programs are summarized at the beginning of the description of RESTORE2 (see "Differences between RESTORE and RESTORE2).

The RESTORE program normally resides in the file named $SYSTEM.SYS<nn>.RESTORE, and the RESTORE2 program resides in the file named $SYSTEM.SYS<nn>.RESTORE2. (SYS<nn> is the subvolume containing the GUARDIAN operating system image currently in use, and <nn> is a two-digit octal integer.)

You can enter RESTORE or RESTORE2 commands in either of two ways:

- Through the GUARDIAN operating system command interpreter (COMINT), by typing a RESTORE or RESTORE2 command at the COMINT colon prompt.
- Through the RESTORE or RESTORE2 programs, by typing the name of the program at COMINT's colon prompt. This starts a RESTORE or RESTORE2 process. Then enter the parameters of your command at the question mark (?) prompt. Type CTRL/Y to exit from RESTORE or RESTORE2.

With either of these methods, you can enter RESTORE or RESTORE2 command parameters through a disc file, as explained in the IN <file-name> description.

See "Mounting a BACKUP Tape" or "Mounting a BACKUP2 Tape" at the beginning of this section for instructions on using tapes created with the BACKUP or BACKUP2 programs.

The RESTORE Program (Display Form)

The command syntax for the display form of RESTORE is given first in this section, followed by the syntax for the restore form. The syntax of the command to read and list the files on a tape without restoring any data is:

```
RESTORE [ \<system-name>.]$<tape-file>  [ , , VERIFYTAPE ]

[ \<system-name>.]$<tape-file>

   See the syntax entry for RESTORE (restore form).

VERIFYTAPE

   verifies the volume label, file label, and data records
   of the files on $<tape-file> without restoring files to
   disc.
```

Listing Format

Figure 4-3 shows the listing format for the RESTORE command. When you run a RESTORE process, the program first displays a one-line program banner at the OUT <list-file>. Then it echoes the IN <file-name> (if you included one in your RESTORE command) and displays its listing as shown in Figure 4-3.

Both the display and restore forms of RESTORE generate a listing. You can direct the listing to a device or file by including the OUT option in your RESTORE command. By default, the listing is sent to the OUT file in effect for the current COMINT (usually this is the home terminal).

```
VOLNAME SVOLNAME FILENAME REEL ERROR ADDRESS <dumpdate> <vsn> [PART]
   .       .        .     .                    <comment>
   .       .        .     .                       .
   .       .        .     .                       .

FILES RESTORED=<num>   FILES NOT RESTORED=<num>   [DATA ERRORS=<num>]
```

Figure 4-3. RESTORE Program Listing Format

The headers and variables that appear in the listing format in Figure 4-3 are:

VOLNAME SVOLNAME FILENAME — is a list of the files specified to the RESTORE program. The file name is indented one space if the file is a secondary partition, or if it is a primary partition and you included the PARTONLY option in your RESTORE command.

REEL — indicates the magnetic tape reel to which the file was dumped. The first reel used in a backup operation is reel one. Files dumped across multiple reels are listed once for each reel.

ERROR — is any file-system error that prevents the normal restoration of a file. Error numbers associated with errors on tape are preceded by a T (for tape). Two tape errors are:

- T011 — File specified in <fileset-list> is not on tape.
- T021 — Bad file format on tape.

ADDRESS — is the byte address of a sector relative to file byte zero.

<dumpdate> — is the date and time of day when the backup operation was started.

<vsn> — is the version level of the operating system at the time the backup operation was performed.

PART	means that the PARTONLY option was included in the command when the tape was written.
<comment>	is a message that may be listed on <list-file> for individual files. Possible comments are:
FILE OPEN	warns you that the OPEN option was specified, and the listed file was open when dumped.
DATA BAD	warns you that you included the IGNORE option, and the sector indicated by ADDRESS contains invalid data (sector = 512 bytes).
DATA LOST	warns you that you included the IGNORE option and the sector indicated by ADDRESS contains invalid data (sector = 512 bytes).
NOT RESTORED	means that the indicated file was not restored because of the file-system error listed under ERROR on <list-file>.
SET ABORTED	means that the remainder of the file set associated with the indicated file was not restored because of the file-system error listed under ERROR on the <list-file>.
FILES RESTORED=<num>	gives the number of files restored to disc.
FILES NOT RESTORED=<num>	gives the number of specified files not restored.
DATA ERRORS=<num>	gives the number of sectors that could not be read or written successfully (is displayed only when the IGNORE option is specified).

Examples of RESTORE (Display Form)

This command requests that RESTORE verify the tape on $TAPE1 and list the files, without writing the tape's contents to disc:

```
:RESTORE $TAPE1, , VERIFYTAPE

GUARDIAN FILE RESTORE PROGRAM - T9074B00 - (18MAR85) SYSTEM \TS
TAPE: $TAPE1              SYSTEM:  \TS      VERSION: B00
VOLNAME SVOLNAME FILENAME REEL ERROR ADDRESS 23 APR 85 10:13

$DATA   RECDS    APRIL    01
$DATA   RECDS    JAN      01
$DATA   RECDS    MAR      01
$DATA   RECDS    FEB      01

FILES RESTORED=0000    FILES NOT RESTORED=0004
```

The next command lists the files on $TAPE2 without verifying them or writing them to disc:

```
:RESTORE $TAPE2
```

The RESTORE Program (Restore Form)

This subsection describes the syntax of RESTORE as it is used for restoring files from tape to disc. The syntax of the command to start a RESTORE process is:

```
RESTORE [ / <run-option> [ , <run-option> ]... / ]
        [ \<system-name>.]$<tape-file> [ , <fileset-list> ]

   [ , AUDITED   [ , TURNOFFAUDIT ]                    ]
   [ , IGNORE                                          ]
   [ , KEEP                                            ]
   [ , LISTALL                                         ]
   [ , MYID                                            ]
   [ , NOT <not-fileset-list>                          ]
   [ , NOUNLOAD                                        ]
   [ , OPEN                                        ... ]
   [ , PARTOF <volume-spec>                            ]
   [ , PARTONLY                                        ]
   [ , REBUILD                                         ]
   [ , START <fileset-name>                            ]
   [ , TAPEDATE                                        ]
   [ , VERIFY                                          ]
   [ , VOL  { [ $<new-vol>.]<new-svol> }               ]
   [        { $<new-vol>               }               ]
```

<run-option>

can be any <run-option> for the RUN command in COMINT. Two common run options are:

IN <file-name>

names a file from which the RESTORE program takes all RESTORE parameters (those parts of the RESTORE command syntax that follow the slash at the end of the list of <run-option>s). If you include the IN option, <file-name> must contain all the RESTORE command parameters that you want to include.

<file-name> can be the name of a disc file, any readable device such as a terminal, or a process.

→

For more information about input files, see the IN <file-name> description for the BACKUP command.

OUT <list-file>

specifies a file to receive all listing and error message output. If you omit the OUT option, RESTORE sends output to the OUT file in effect for the current COMINT process. (If you are using COMINT interactively, COMINT's OUT file is your home terminal.)

If you specify a <list-file> that does not already exist, RESTORE creates an entry-sequenced disc file named <list-file> with a record length of 80 characters, block size of 2048 bytes, primary extent size of 4 pages, and secondary extent size of 16 pages.

See the RUN command in COMINT (Section 2) for a complete <run-option> list.

[\<system-name>.]$<tape-file>

specifies the magnetic tape unit from which the files are to be restored.

\<system-name>

is the name of the system where $<tape-file> resides. If you omit \<system-name>, RESTORE assumes that $<tape-file> resides in the system where you entered the RESTORE command.

$<tape-file>

is the name of the tape unit to be used. For $<tape-file>, specify either of these:

$<device-name>
$<logical-device-number>

→

(For more detail, see the description of $<tape-file> in the BACKUP command description.)

<fileset-list>

designates one or more sets of related files to be restored. For <fileset-list>, you can specify either of these:

```
<fileset>
( <fileset> [ , <fileset> ]... )
```

<fileset>

is a set of files to be restored. For <fileset>, specify any one of these four:

```
[ $<volume>.][<subvol>.]<file-name>
[ $<volume>.][<subvol>.]*
[ $<volume>.]*.*
*.*.*
```

(For more details, see the description of <fileset> in the BACKUP command description.)

If you omit <fileset>, RESTORE sends a list of all the files in the tape set to the OUT <list-file> using the LISTALL format, but does not restore any files. (The listing format is presented at the end of the syntax for the display form of RESTORE.)

AUDITED

restores audited files. Files audited by TMF at the time they were backed up can be restored only by including the AUDITED option.

→

IGNORE

ignores data errors occurring on tape. Retries data errors on a sector-by-sector basis (sector = 512 bytes). If you do not include IGNORE and a data error occurs, RESTORE purges the current file and then proceeds with the next file.

KEEP

keeps any file on disc that has the same name as a file being restored from tape (the backup file on tape does not replace the file on disc). If you omit KEEP, the file on disc is replaced with the file on the tape.

LISTALL

lists the names of all files restored from <fileset-list> as well as any errors that occur. If you omit LISTALL, RESTORE lists only those file names associated with error messages.

MYID

sets the "owner ID" of all files being restored to that of the user who runs RESTORE. Each file is given the user's current default security. If MYID is omitted, the owner IDs and file security settings are set to the values in effect when the files were backed up.

NOT <not-fileset-list>

is a list of files that are not to be included in the restoration. The form of <not-fileset-list> is the same as for <fileset-list>.

→

NOUNLOAD

means that the final tape is rewound and left online when the restore operation completes. If you omit NOUNLOAD, the last tape (like all the preceding tapes) is rewound and unloaded when the restoration completes.

OPEN

restores files even if they are open at the time of backup (that is, BACKUP with the OPEN option). Omitting OPEN means that files of this type are not restored.

PARTOF <volume-spec>

restores an individual partition of a partitioned file. PARTOF restores any partition in the <fileset-list> whose primary partition resides on the volume defined by <volume-spec>; nonpartitioned files defined by <fileset-list> are not restored. If this option is specified, the <fileset-list> must define a single volume. Moreover, the PARTONLY option of BACKUP must not have been specified when the tape was made. If it was, the restore operation is aborted.

<volume-spec>

specifies where a partition's primary file must reside if the partition is to be restored. It is either:

$<volume-name> or *

If you give a $<volume-name>, only the partitions whose primary partition resides on $<volume-name> and that are defined by <fileset-list> are restored.

If you give "*", then any partitions defined by the <fileset-list> are restored.

If you omit PARTOF and if the PARTONLY option of BACKUP was not specified when the tape was made, the entire

→

partitioned file of any primary partition defined in the <fileset-list> is restored, and secondary partitions defined by the <fileset-list> are not restored.

PARTONLY

restores individual primary or secondary partitions defined in the <fileset-list> (entire partitioned files as a whole are not restored). To use PARTONLY with RESTORE, you must have used PARTONLY with BACKUP when the tape was made (the listing of the tape contents indicates whether PARTONLY was used at backup). If PARTONLY was not used, RESTORE does not restore any partitions. To restore individual primary and secondary partitions from a tape made without PARTONLY, use the PARTOF option.

REBUILD

means that the files to be restored on the volumes in <fileset-list> are defined by the file directories on option are usually created using the PARTIAL option of the BACKUP program. Start your RESTORE with the most current tape set first. If any files defined in the directory on the first tape set do not exist on that tape set, RESTORE prompts you to mount the preceding (older) tape set. This process of restoring files from preceding tape sets continues until either all files are restored, or you terminate the RESTORE operation by typing STOP when RESTORE prompts you for another tape. Any files on the volumes before the REBUILD begins are retained (REBUILD implies KEEP).

START <fileset-name>

indicates where in <fileset-list> RESTORE is to begin. The START option works for RESTORE as it does for BACKUP, except that you cannot use the wild-card asterisk (*) in <fileset-list> with RESTORE.

→

Considerations

- If you include an IN file in your RESTORE command, the IN file can contain over 100 file sets that are a part of either <fileset-list> or <not-fileset-list>.

- If a program file to be restored has the PROGID attribute set on (see the FUP commands SECURE and REVOKE), the file retains the setting only if it is restored by the super ID user or the owner of the file. Similarly, a licensed file loses its license if it is restored by a user other than the super ID user (see the FUP commands LICENSE and REVOKE).

- If you do not include the KEEP option and a file exists on disc with the same name as a file to be restored, the file on disc is purged and replaced with the file on the backup tape.

- If a partitioned file is included in the <fileset-list> and the PARTONLY and PARTOF options are not specified, all partitions are automatically restored (if they are all on the backup tape). If, however, the tape contains a secondary partition whose primary partition is on an earlier reel, the secondary partition is skipped.

- If you include the PARTOF option, RESTORE must read the tape set until it finds the primary partition before it can restore any individual secondary partitions.

- Files restored using the TAPEDATE option receive the modification timestamp in effect when they were backed up. Therefore, if you later want to back up this file using the PARTIAL option, the file is not backed up if the date in its timestamp is earlier than the <partial-dump-date>.

- RESTORE checks tapes for checksum errors and reports these errors whether or not you included the VERIFYTAPE option in the BACKUP command when the files were backed up.

- The checksum error messages for the BACKUP command also apply to the RESTORE command. If you do not use the IGNORE parameter and RESTORE encounters a checksum error during the restore, the following message is displyed at your terminal:

  ENTER CR TO CONTINUE, ENTER STOP TO ABORT

  To continue the restore process, with the bad checksum record written to disc, press RETURN. To abort the restore process, enter "stop".

If you include the IGNORE parameter, RESTORE writes any bad checksum records to disc.

- The RESTORE program restores backup tapes written by the BACKUP program that was available before the Tandem April 1, 1982, software release, as well as tapes produced by the BACKUP program available after that release. RESTORE also restores system image tapes written in both the old and new formats.

- To restore tape files that were backed up using the BACKUP2 program, you might have to use RESTORE2. You can use RESTORE if files were backed up with BACKUP2 with the OLDFORMAT or DP1FORMAT options, or if only DP1 files were backed up with BACKUP. (That is, RESTORE can read only tape version 0 or 1.)

Considerations for Files Audited by TMF

- If a file was audited by TMF when it was backed up, you must include the AUDITED option in the RESTORE command when restoring it. This is also true of audited files that were backed up before the December 1983 software release, when the AUDITED option was included in the BACKUP and RESTORE commands. TMF has its own recovery mechanisms for audited files, and you should not use BACKUP and RESTORE in place of them, except in extraordinary circumstances.

  You might, however, want to use BACKUP and RESTORE for these purposes:

  --For compressing data in a disc volume containing audited files (See your System Operator's Guide for information about disc compression.)

  --For transporting audited files to another system

  --In the event of a TMF emergency

  --For archiving files when the RECOVERY OFF option of TMF is used

  If you do not include the AUDITED option, and RESTORE encounters a file that was previously audited, RESTORE skips the file and sends the message *** AUDITED FILE SKIPPED to the output file.

- TMF must be running when you run RESTORE if you want previously audited files to be audited when you restore them. Note, however, that if you include TURNOFFAUDIT with the AUDITED option in the RESTORE command:

  --The files that are restored are not audited even if TMF is running when you run the RESTORE program

  --The message *** AUDIT TURNED OFF is sent to the output file for each file that was audited but not open when it was backed up; the message OPEN, AUDIT TURNED OFF is sent for each file that was both open and audited when it was backed up

- If, because TMF was not running, a file that was audited when it was backed up is not audited when it is restored, RESTORE sends the message *** WARNING: COULDN'T MAKE FILE AUDITED to the output file.

- If a file was audited when it was backed up, you cannot RESTORE it if:

  --A file with the same name exists on disc and is audited

  --TMF is configured, but is not currently running

  When such an attempt fails, the message *** FILE SKIPPED is sent to the output file, and the file is skipped. If this happens, you can restore the file by specifying a different subvolume name for it with the VOL option.

- When you are restoring a file that already exists, RESTORE normally purges the old file first. If the file is an audited file, purging causes all corresponding dump entries to be deleted from the TMF catalog.

Listing Format

The listing format for the RESTORE program is illustrated and described following the syntax desciption for the display form of RESTORE (see Figure 4-3). Examples of RESTORE listings also follow the syntax for the display form of RESTORE.

Examples of RESTORE (Restore Form)

This command requests that RESTORE verify the tape in device $TAPE and write its contents to disc. The same volume, subvolume, and file names are used on disc and tape:

```
:RESTORE $TAPE , *.*.*
```

The following command is illegal because it combines the two forms of RESTORE. You cannot include a <fileset-list> in the same RESTORE command with VERIFYTAPE:

```
:RESTORE $TAPE , *.*.* , VERIFYTAPE
```

This command restores all files on the backup tape for all volumes in the system except those for the volume $SYSTEM. The KEEP option means that if any file on disc has the same name as a file on the backup tape, the copy on the backup tape is ignored and the file on disc is retained:

```
:RESTORE $TAPE2, *.*.*, NOT $SYSTEM.*.*, KEEP, LISTALL
```

This command restores all files on tape that have the volume and subvolume name $STORE2.ACCTSRCV to the subvolume ACCTSRCV on the volume $STORE1. Any file on $STORE1.ACCTSRCV that has the same name as a file on the backup tape is purged and the backup copy is restored. The file names of associated files are listed on the OUT file or device:

```
:RESTORE $TAPE, $STORE2.ACCTSRCV.*, VOL $STORE1, LISTALL
```

This command sequence restores all files and partitions of files for the volume $STORE1. The PARTONLY option was specified in the BACKUP command that created this tape:

```
:VOLUME $STORE1
:RESTORE $TAPE, *.*, PARTONLY, LISTALL
```

The following command sequence restores all partitions of files on $SYSTEM whose primary partitions reside on the volume $STORE1. The PARTONLY option was not specified when the backup operation was performed:

```
:VOLUME $SYSTEM
:RESTORE $TAPE3, *.*, PARTOF $STORE1, LISTALL
```

Differences between the RESTORE and RESTORE2 Programs

You can use either the RESTORE or the RESTORE2 program to restore backup files from tape to disc. RESTORE can work with tapes in either of two formats--the old backup-format or DP1 format. For this reason, RESTORE can read any tape created with the BACKUP program, but it can read only those BACKUP2 tapes that are in tape version 0 or tape version 1 (that is, BACKUP2 tapes created with the OLDFORMAT option or with the DP1FORMAT option).

RESTORE2, on the other hand, can work with any tape created by BACKUP or BACKUP2--that is, any of the following tape formats:

- Old backup-format (BACKUP2 tape version 0 or any BACKUP tape created with the OLDFORMAT option)
- DP1 format (BACKUP2 tape version 1 or any BACKUP tape in default format)
- DP2 format (BACKUP2 tape version 2)

The basic function of RESTORE2 is the same as that of RESTORE, but RESTORE2 has expanded capabilities, and its listing format is similar to that of BACKUP2. RESTORE2 has these additional command parameters:

- EXT sets the primary and secondary extent sizes for the destination file in a file conversion (when converting files from one disc-process type to another).
- DSLACK sets the minimum percentage of slack space in data blocks in a file conversion.
- ISLACK sets the minimum percentage of slack space in index blocks in a file conversion.
- PART renames the secondary partitions of partitioned files and can restore secondary partitions from tape to a new volume.
- ALTFILE changes the names of alternate-key files in the file label of the primary-key file.

You can also use RESTORE2 with the VOL option to restore files to remote systems.

The RESTORE2 Program (Display Form)

The command syntax for the display form of RESTORE2 is given first in this section, followed by the syntax for the restore form. The syntax of the command to read and list the files on a tape without restoring any data is:

```
RESTORE2 [ \<system-name>.]$<tape-file>  [ , , VERIFYTAPE ]
```

[\<system-name>.]$<tape-file>

is the name of the tape drive that contains the tape you want to read. See the syntax entry for RESTORE.

VERIFYTAPE

verifies the volume label, file label, and data records of the files on $<tape-file> without restoring files to disc.

Listing Format

Figure 4-4 shows the listing format for the RESTORE2 command. When you run a RESTORE2 process, the program first displays a one-line program banner at the OUT <list-file>. Then it echoes the IN <file-name> (if you included one in your command) and displays a listing as shown in Figure 4-4.

Both the display and restore forms of RESTORE2 generate a listing. You can direct the listing to a device or file by including the OUT option in your RESTORE2 command. By default, the listing is sent to the OUT file in effect for the current COMINT (usually this is the home terminal).

```
[*WARNING*   Old format tape -- no checksums.]
Tape: [\<system.]$<tape>  Operating System: <vsn>  Tape Version: <v>
Backup options: <params>
[Partial Time: <date> <time> ]
Restore [(list only)] time:<date><time>  Backup time:<date><time> Page:<n>

Reel: <n>      Code     EOF   Last modif  Owner  RWEP  Type   Rec  Block

$<vol>.<subvol>
 <filename>    <code>  <eof> <date><time>  <g,u> <rwep><type> <rec> <bl>
    .    .       .       .     .      .      .     .
    .    .       .       .     .      .      .     .
[            *ERROR*    <comment>          ]
[            *WARNING*  <comment>          ]

[ Files restored = <n>  Files not restored = <n> ] [ Data errors = <n> ]
[ Files on tape = <n> ]
```

Figure 4-4. RESTORE2 Program Listing Format

The listing headers and variables that appear in Figure 4-4 are:

WARNING Old format tape -- no checksums.

appears when you run RESTORE2 on a tape containing files in the old backup-format, in which checksums are not calculated.

TAPE: [\system.]$<tape> is the name of the tape drive where the tape containing the listed files is mounted.

Operating System:<vsn> is the version level of the operating system at the time the backup operation was performed.

Tape Version: <v> identifies the version of the tape format. <v> can be 0 (for old format), 1 (for DP1 format), or 2 (for DP2 format).

Backup options: <params> lists all the command parameters that might affect the RESTORE2 operation for these files. The tape format determines the options that are listed.

	That is, if the tape is in the old format, only one option is listed ([NO] PARTONLY). For a DP1-format tape, one additional option appears (BLOCKSIZE <n>). For a DP2-format tape, three more options appear ([NO] AUDITED, [NO] OPEN, and [NO] IGNORE). For details, see "Considerations."
Partial Time:<date><time>	is the date and time you specified with the PARTIAL option in your BACKUP2 command. This information appears only when you restore a tape of tape-version 2 (that is, a tape of DP2 files).
Restore [(list only)] time:<date><time>	gives the date and time when the restore operation began. If "(list only)" appears, this RESTORE2 operation only listed the contents of the tape and did not restore any files to disc.
Backup time:<date><time>	gives the date and time when the backup operation began.
Page: <n>	is the page number of the listing.
Reel: <n>	indicates the magnetic tape reel whose files are being restored. The first reel used in a backup operation is reel one. Files dumped across multiple reels are listed once for each reel.
$<vol>.<subvol>	is the volume and subvolume name of the files being restored.
<file-name>	is the file name of each file in the file set to be restored.
CODE <code>	is the file code of the listed file. Reserved file codes are listed in Table 3-1 in the FUP section.
EOF <eof>	is the end-of-file value (the size of the file in bytes).
Last modif <date><time>	is the date and time that the listed file was last modified.
Owner <g,u>	is the group ID and user ID of the owner of the listed file.

RWEP <rwep>	is the security of the listed file.
Type <type>	is the file type of the listed file. <type> can be any of these: <null> = unstructured file R = relative file E = entry-sequenced file K = key-sequenced file <t>A = file with alternate key P<t> = partitioned file XP<t> = file is an extra partition (secondary partition) where <t> can be one or more of R, E, K, P, or X.
Rec <rec>	for ENSCRIBE structured files, is the file's logical record length in bytes. This field is blank for unstructured files.
Block <bl>	for ENSCRIBE structured files, is the file's block length in bytes. This field is blank for unstructured files.
ERROR <comment>	is any file-system error that prevents the normal restoration of a file.
WARNING <comment>	is a warning about the listed file.
Files restored = <n>	is the number of files written to disc.
Files not restored = <n>	is the number of files that were specified for restoration but were not written to disc.
Files on tape = <n>	is the number of files that are on this tape reel. This information appears only when you run RESTORE2 for display purposes only.
Data errors = <n>	is the number of data errors encountered in the files that were restored.

Examples of RESTORE2 listings follow the syntax for the restore form of RESTORE2 later in this section.

The RESTORE2 Program (Restore Form)

This subsection describes the syntax of RESTORE2 as you use it to restore files from tape to disc. The syntax description includes only those parameters that are unique to RESTORE2; parameters that are the same for both RESTORE2 and RESTORE are described in the syntax for RESTORE, as indicated here by the note, "See the syntax entry for RESTORE."

The syntax of the command to restore files from tape to disc is:

```
RESTORE2 [ / <run-option> [ , <run-option> ]... / ]
  [ \<system-name>.]$<tape-file>  [ , <fileset-list> ]

  [, ALTFILE ( <key-file-number> ,                                  ]
  [          [$<vol>.][<subvol>.]<file-name> )                      ]
  [, AUDITED   [ , TURNOFFAUDIT ]                                   ]
  [, DSLACK <percentage>                                            ]
  [, EXT  {<extent-size>                                           } ]
  [       {( <pri-extent-size> , <sec-extent-size ) }              ]
  [, IGNORE                                                         ]
  [, ISLACK <percentage>                                            ]
  [, KEEP                                                           ]
  [, LISTALL                                                        ]
  [, MYID                                                           ]
  [, NOT <not-fileset-list>                                         ]
  [, NOUNLOAD                                                       ] ...
  [, OPEN                                                           ]
  [, PART ( <secondary-partition-num>                               ]
  [       , [ [ \<system-name>.]$<volume-name> ]                    ]
  [         [ , <pri-extent-size> , [ <sec-extent-size> ]] )        ]
  [, PARTOF <volume-spec>                                           ]
  [, PARTONLY                                                       ]
  [, REBUILD                                                        ]
  [, START <file-name>                                              ]
  [, TAPEDATE                                                       ]
  [, VERIFY                                                         ]
  [, VOL  {[\<system-name>][ $<new-vol>.]<new-svol>}                ]
  [       {[\<system-name>]$<new-vol>              }                ]
```

→

<run-option>

can be any <run-option> for the RUN command in COMINT. Two common <run-option>s are:

IN <file-name>

names a file from which RESTORE2 takes all command parameters (those parts of the RESTORE2 command syntax that follow the slash at the end of the <run-option> list). If you include the IN option, <file-name> must contain all the RESTORE2 command parameters that you want to include.

<file-name> can be the name of a disc file, a process, or any readable device such as a terminal.

For more information, see the IN <file-name> description for the BACKUP command.

OUT <list-file>

specifies a file to receive all listing and error message output. If you omit the OUT option, RESTORE2 sends output to the OUT file named in the command that started the current COMINT process. (If you are using COMINT interactively, COMINT's OUT file is your home terminal.)

If you specify a <list-file> that does not already exist, RESTORE2 creates an entry-sequenced disc file named <list-file> with a record length of 80 characters, block size of 2048 bytes, primary extent size of 4 pages, and secondary extent size of 16 pages.

See the RUN command in COMINT for more <run-option>s.

[\<system-name>.]$<tape-file>

specifies the magnetic tape unit to which the files are to be dumped.

→

\<system-name>

names the system where $<tape-file> resides. If you omit \<system-name>, RESTORE2 assumes that $<tape-file> resides in the system where you entered the RESTORE2 command.

$<tape-file>

is the name of the tape unit to be used. The form of $<tape-file> is:

```
$<device-name>
$<logical-device-number>
```

(For details, see the description of $<tape-file> in the BACKUP command description.)

<fileset-list>

designates one or more sets of related files to be restored. The form of <fileset-list> is:

```
<fileset>
( <fileset> [ , <fileset> ]... )
```

<fileset>

is a set of files to be restored. The form of <fileset> is:

```
[\<system>.][ $<volume>.][<subvol>.]<file-name>
[\<system>.][ $<volume>.][<subvol>.]*
[\<system>.][ $<volume>.]*.*
[\<system>.]*.*.*
```

(For more details, see the description of <fileset> in the BACKUP command description.)

If you omit <fileset>, RESTORE2 lists all the files in the tape set to the OUT <list-file> in the LISTALL format, but it does not restore any files (see the syntax description for the display form of RESTORE2).

```
ALTFILE ( <key-file-number> ,
        [$<vol>.][<subvol>.]<file-name> )
```

changes the name for an alternate-key file in the file label of the primary-key file. Specify <key-file-number> as an integer in the range from 0 to 255, inclusive, designating the alternate-key file you are naming. For $<vol>.<subvol>, specify the volume and subvolume where the alternate-key file is to reside (if they differ from the file's original volume and subvolume). For <file-name>, give the name of the alternate-key file.

AUDITED

See the syntax entry for RESTORE.

DSLACK <percentage>

used when converting key-sequenced files from one disc-process type to another; sets the minimum percentage of slack space in data blocks. If you omit this option, RESTORE2 uses the SLACK percentage value (see the LOAD command in FUP).

```
EXT  {<extent-size>                                  }
     {( <pri-extent-size> , <sec-extent-size> )      }
```

used when converting files from one disc-process type to another; sets the extent size of the file copy to be made on disc. For <extent-size>, <pri-extent-size>, and <sec-extent-size>, specify an integer in the range from 1 to 65,535, inclusive, designating the extent size in pages (units of 2048 bytes).

IGNORE

See the syntax entry for RESTORE.

→

ISLACK <percentage>

used when converting key-sequenced files from one disc-process type to another; sets the minimum percentage of slack space in index blocks. If you omit this option, RESTORE2 uses the SLACK percentage value (see the LOAD command in FUP).

KEEP

See the syntax entry for RESTORE.

LISTALL

See the syntax entry for RESTORE.

MYID

See the syntax entry for RESTORE.

NOT <not-fileset-list>

See the syntax entry for RESTORE.

NOUNLOAD

See the syntax entry for RESTORE.

OPEN

See the syntax entry for RESTORE.

```
PART ( <secondary-partition-num>
     , [ [ \<system-name>.]$<volume-name> ]
     [ , <pri-extent-size> , [ <sec-extent-size> ] ] )
```

specifies a new name for a secondary partition of a partitioned file. The PART option causes the secondary-partition name to be inserted in the file label of the primary partition. The PART parameters <secondary-partition-num>, \<system-name>, and <volume-name> are described in the SET command in the FUP section (Section 3).

If you specify PART, the original name of the partition indicated by <secondary-partition-num> is replaced by the new partition name you specify. The <secondary-partition-num> should already exist in the source file.

If you omit the volume name, RESTORE2 uses the original volume name of the secondary partition.

You can also optionally specify the destination partition extent sizes. Specify <pri-extent-size> and <sec-extent-size> as integers in the range from 1 to 65,535, inclusive, indicating the size of the primary and secondary extents, respectively, in units of pages (1 page is 1024 bytes). RESTORE2 uses the extent sizes only when converting files from one disc-process type to another.

If you include both PART and PARTONLY, RESTORE2 changes the name of the secondary partition in the file label of the primary partition. If you include PART without PARTONLY, RESTORE2 changes the secondary-partition name in the primary-partition file label and changes the actual file name of the secondary partition as well.

```
PARTOF <volume-spec>
```

is the same for RESTORE2 as for RESTORE, except that <volume-spec> for RESTORE2 can be any of these three:

```
*
$<volume-name>
( $<volume-name> , $<volume-name> ... )
```

→

With RESTORE2, the <fileset-list> can define more than one volume, and the PARTOF option can also define multiple volumes.

PARTONLY

See the syntax entry for RESTORE.

REBUILD

See the syntax entry for RESTORE.

START <file-name>

See the syntax entry for RESTORE.

TAPEDATE

See the syntax entry for RESTORE.

TURNOFFAUDIT

See the syntax entry for RESTORE.

VERIFY

See the syntax entry for RESTORE.

```
VOL  { [ \<system-name>.][ $<new-vol>.]<new-svol> }
     { [ \<system-name>.]$<new-vol>               }
```

works for RESTORE2 as for RESTORE, except that you can cause RESTORE2 to convert files from DP1 format to DP2 format (or vice versa) by specifying a $<new-vol> that is a different disc-process type from the original volume of the files being restored. To restore files to a remote system, specify the name of the remote system for \<system-name>.

→

CAUTION

If you specify a new volume or subvolume name with the VOL option so that the same file name is given to two or more files being restored, RESTORE2 restores only the last file.

Considerations

- You can use RESTORE2 in the following cases:

  --To restore files from any tape created with the BACKUP program (RESTORE2 can restore files from a DP1-format or old-format BACKUP tape; the restored files can be written in either DP1 format or DP2 format.)

  --To restore files from any tape created with BACKUP2 (RESTORE2 can restore files from BACKUP2 tapes in version 0, 1, or 2. The restored files are written in the same format as the backup tape; you can, however, convert files from one format to another by using the VOL option to restore files to a disc volume of another disc-process type).

- You must use RESTORE2 rather than RESTORE to restore any tape of tape version 2 (DP2 files), a format created only by BACKUP2.

- The listings for RESTORE2 and BACKUP2 both include "Backup options". The options listed vary according to the tape version created and according to the command options that you include in your BACKUP2 command. In general, the options in your RESTORE2 command must match those in the BACKUP2 command that dumped these files to tape. The following options are particularly significant:

  --The PARTONLY specification must be the same for BACKUP2 and RESTORE2 if you are restoring partitioned files. That is, if NO PARTONLY appears in the listing, you cannot include the PARTONLY option in your RESTORE2 command.

--The BLOCKSIZE specification must be 2 (and the tape version must be 0 or 1) in order to restore the files to a NonStop 1+ system. The default BLOCKSIZE for NonStop systems is 8 (8192 bytes).

--The AUDITED and OPEN backup options indicate whether the tape might contain audited files and files that were open files at the time of backup.

- When RESTORE2 converts files from one format to another, it creates temporary disc files on the current default subvolume. The temporary files, which are in the destination-file format, have names that begin with ZZRA. If you terminate RESTORE2 early, the default subvolume may contain some temporary files. To save disc space, purge any remaining ZZRA files.

Listing Format

The listing format for the RESTORE2 program is presented after the syntax description of the display form of RESTORE2 (see Figure 4-4).

Examples

This example restores all the files on the backup tape created in the first example given for BACKUP2. These files were originally DP1 files; they were converted in the BACKUP2 operation to DP2 format (tape version 2). This RESTORE2 command restores the files to a DP2 disc volume:

```
:RESTORE2 $TAPE2, *.*.*, LISTALL
```

```
Guardian File RESTORE2 Program - T9073B00 - (18MAR85)   System: \TS
Tape: $TAPE2  Operating System: B00  Tape Version: 2
Backup options: NO AUDITED, BLOCKSIZE 8, NO IGNORE, NO OPEN, NO PARTONLY
Restore time:  5Jan85  9:48  Backup time:  9Nov84 17:47                Page: 1

Reel: 1        Code           EOF    Last modif    Owner RWEP   Type   Rec Block

$DP2.RECDS
 ENTALT       *WARNING*  At least one alternate key file has a different volume
                         type (Dp1/Dp2).
 ENTALT                    62464  9Nov84 14:37   1,34  CUCU   E       80  1024
 ENTALT1                   82944  9Nov84 14:37   1,34  CUCU   K       27  1024

$DP2.RECDS
 ENTPART      *ERROR*    File aborted. Not all the partitions of this entire
                         partitioned file have the same volume type (Dp1/Dp2).

$DP2.RECDS
 KEYALT       *WARNING*  At least one alternate key file has a different volume
                         type (Dp1/Dp2).

 KEYALT                    62464  9Nov84 14:37   1,34  CUCU   K       80  1024
 KEYALT1                   72704  9Nov84 14:38   1,34  CUCU   K       19  1024
 KEYALT2                   13312  9Nov84 14:38   1,34  CUCU   K       27  1024

$DP2.RECDS
 RELALT       *WARNING*  At least one alternate key file has a different volume
                         type (Dp1/Dp2).
 RELALT                   279552  9Nov84 14:38   1,34  CUCU   R       80  102
 RELALT1                   72704  9Nov84 14:38   1,34  CUCU   K       19  1024
 RELALT2                   13312  9Nov84 14:38   1,34  CUCU   K       27  1024

$DP2.RECDS
 RELPART      *ERROR*    File aborted. Not all the partitions of this entire
                         partitioned file have the same volume type (Dp1/Dp2).

Files restored = 8  Files not restored = 2
```

This RESTORE2 listing gives warnings and errors that indicate the following:

- For ENTALT, KEYALT, and RELALT (structured files of different types with alternate-key files), pointers in the primary file still point to files with DP1 volume names. To correct this situation, you must rename each restored alternate-key file to the correct volume type (in this case, a volume in DP2 format, such as the volume used in this example, $DP2). You can do this by entering a FUP ALTER command with the ALTFILE option.

- For ENTPART and RELPART (partitioned files), the primary partitions have been restored to a DP2 volume in the system ($DP2). However, the volume names of the secondary partitions are the names of the original DP1 volumes. Because of this discrepancy in volume names for the primary and secondary partitions, none of the secondary partitions are restored.

To correct this situation, you must incude the PART option of RESTORE2 to rename each secondary partition to a volume of the same disc-process type as the primary partition. Enter a RESTORE2 command with the PART option, like this:

```
:RESTORE2 $TAPE2, $DP2.RECDS.KEYPART, PART (1, $DP2A), &
:PART (2, $DP2B), NOUNLOAD, LISTALL
```

The next example demonstrates using RESTORE2 to convert all the files on a DP1-format tape (tape version 1) to DP2 disc files. (The VOL option names a DP2 volume to contain the restored files.)

```
:RESTORE2 $TAPE3, *.*.*, NOUNLOAD, LISTALL, VOL $DP2VOL.FILES
```

```
Guardian File RESTORE2 Program - T9073B00 - (18MAR85)   System: \TS
Tape: $TAPE3  Operating System: B00  Tape Version: 1
Backup options: BLOCKSIZE 8, NO PARTONLY
Restore time:  9Feb85 17:48  Backup time:  9Nov84 17:47                Page: 1

Reel: 1        Code           EOF    Last modif    Owner RWEP   Type    Rec Block

$DP2VOL.FILES
 PAYROLL      *WARNING*  At least one alternate key file has a different volume
                         type (Dp1/Dp2).
 PAYROLL                    62464  9Nov84 14:37    8,44  CUCU   E        80  1024
 PAYALT1                    82944  9Nov84 14:37    8,44  CUCU   K        27  1024
 PAYALT2                    72704  9Nov84 15:02    8,44  CUCU   K        19  1024
 PAYALT3                     1024  9Nov84 17:21    8,44  CUCU   E        80  1024

$DP2VOL.FILES
 PERSONL      *ERROR*    File aborted. Not all the partitions of this entire
                         partitioned file have the same volume type (Dp1/Dp2).

$DP2VOL.FILES
 QRTRS        *ERROR*    File aborted. Not all the partitions of this entire
                         partitioned file have the same volume type (Dp1/Dp2).

Files restored =  4  Files not restored = 2
```

In this RESTORE2 listing, note the following:

- A warning is displayed for the file PAYROLL. Its file-label pointers for the three restored alternate-key files retain the original DP1 volume names, while the primary-key file itself was restored to a DP2 volume named $DP2VOL.
- The partitioned file PERSONL is not restored to disc because its secondary partitions have DP1 volume names.
- The unstructured file QRTRS is not restored because its secondary partitions have DP1 volume names.

To correct the situation that caused the warning for the file PAYROLL, you must change the file-label pointers in the restored primary-key file (PAYROLL) to contain the correct volume name for each restored alternate-key file. To do this, include the ALTFILE option in a FUP ALTER command (see Section 3).

To restore PERSONL and QRTRS, enter a RESTORE2 command with the PART option (see the previous example and the syntax description for RESTORE2).

SECTION 5

PERIPHERAL UTILITY PROGRAM (PUP)

The Peripheral Utility Program (PUP) performs various operations on discs and other peripheral devices connected to the system. For example, you can use PUP to prepare an uninitialized disc pack for use, write a new volume label on a previously formatted disc, prepare an uninitialized disc pack for system use or write a new volume label on a previously formatted disc volume.

This section of the manual contains the detailed syntax, considerations, and examples of each PUP command. Refer to the System Messages Manual for a list of PUP error messages.

For more information about the tasks you can perform using PUP, refer to Section 5 of the System Operator's Guide.

RUNNING PUP

You can use PUP either interactively at an online terminal or noninteractively through commands read from a command file.

Interactive Use

When using the PUP program interactively from the command interpreter, you simply enter the following command:

```
:PUP
```

PUP responds with the following banner and prompt ("#" is the PUP prompt character):

In a normal PUP execution cycle, PUP prompts you for a command, you respond by entering the command, PUP executes the command, and the cycle repeats.

For example, to list system devices and their characteristics, assuming you have entered PUP as shown above, enter the LISTDEV command:

```
#LISTDEV

GUARDIAN PERIPHERAL UTILITY PROGRAM - T9074B00 - (18MAR85)    SYSTEM  \TS
#


LDEV  NAME STATE  PPID   PCU     BPID   BCU  DMNT TYPE SUBTYPE RECSIZE

   0          ON  00,003 %3      04,003             1      0      102
   1  $NCP        00,004         01,003            62      0        0
   3  $OSP        00,005         04,004             6      0       80
   4  $TEST-P  *  00,009 %114    01,007 %114   Y    3      3     4096
      $TEST-B     00,009 %14     01,007 %14
   6  $TAPE       00,010 %30     01,010 %30         4      0     2048
   7  $SLP        00,010 %57     01,010 %57         5      2      120
   8  $LP         00,011 %21     01,011 %21         5      1      132
   9  $CONSOL     00,012 %40     01,012 %40         5      3      132
  10  $TERM0      00,013 %41     01,013 %41         6      1       80
  11  $TERM1      00,013 %42     01,013 %42         6      1       80
   .
   .
   .
  68  $TERM56     00,014 %41     01,014 %41         6      2       80
#
```

To exit from PUP and return to the command interpreter, enter an EXIT command:

```
#EXIT
:
```

You can also start PUP to execute one command and then terminate. To do this, you issue command lines of the following type at the command interpreter prompt:

```
:PUP <command-name> <parameter-string>
```

The following examples compare command lines executed in this way to the corresponding series of commands required for an interactive session.

Command Line	Equivalent To
:PUP LISTDEV :	:PUP #LISTDEV #EXIT :
:PUP LISTBAD $ACCT :	:PUP #LISTBAD $ACCT #EXIT :

Noninteractive Use

To run PUP noninteractively from the command interpreter, you enter a command of the following type:

```
:PUP [ / <run-option> [ , <run-option> ] ... / ]
```

PUP reads from the file specified by the IN <filename> option. PUP directs all listing output to the list device specified by the OUT <filename> option. When PUP reaches the end of file or reads an EXIT command, it terminates, and the command interpreter is reactivated.

Syntax to Run PUP

The PUP program resides in a file designated $SYSTEM.SYSnn.PUP (where SYSnn, with nn a two-digit number, is the currently used system subvolume).

Normally, PUP runs through the GUARDIAN command interpreter. The syntax of the command to run the PUP program, either interactively or noninteractively, is:

```
PUP [ / <run-option> [ , <run-option> ] ... / ]

    [ <command> ]
```

<run-option>

is an option to the GUARDIAN command interpreter RUN command, as described in Section 2 in this manual. The IN <filename> option, if specified, names the file from which PUP is to take its input commands; the OUT <filename> option, if specified, names the file to which PUP is to send its output. The default for both files is the home terminal.

<command>

is a PUP command. If <command> is included, PUP executes it and then terminates.

Considerations

- To perform certain PUP operations, such as a PUP CHECKSUM, on the system disc, you must use a copy of the program file for PUP that resides on a disc other than the system disc.
- While PUP is executing a command that writes to a device, that device is temporarily inaccessible to other programs. During the execution of some PUP commands, the device is placed in an "S" or "R" state (refer to the description of the LISTDEV command later in this section for an explanation of these states). If PUP is aborted when a device is in one these states (for example, by your explicitly stopping PUP), the device is left in that state.
- All numbers entered into or displayed by PUP are treated as base 10 (decimal), unless preceded by a percent sign (%) to indicate octal representation.

Example

To send to printer $HT1 a list of all devices, enter the command:

```
:PUP / OUT $S.#HT1 / LISTDEV
```

DEVICE DESIGNATIONS IN PUP COMMANDS

Nearly all PUP commands refer to an I/O device. To use these references properly, you must understand how both the I/O device configurations and the device references are categorized.

I/O Device Configurations

The I/O device configurations can, for the most part, be separated into two categories: 1) nondisc devices, and 2) dual-port disc volumes (mirrored and nonmirrored). For nondisc devices, the most common configuration has a single dual-port controller controlling one or more devices (see Figure 5-1). Devices in this category include:

- Terminals
- Line printers
- Magnetic tape units

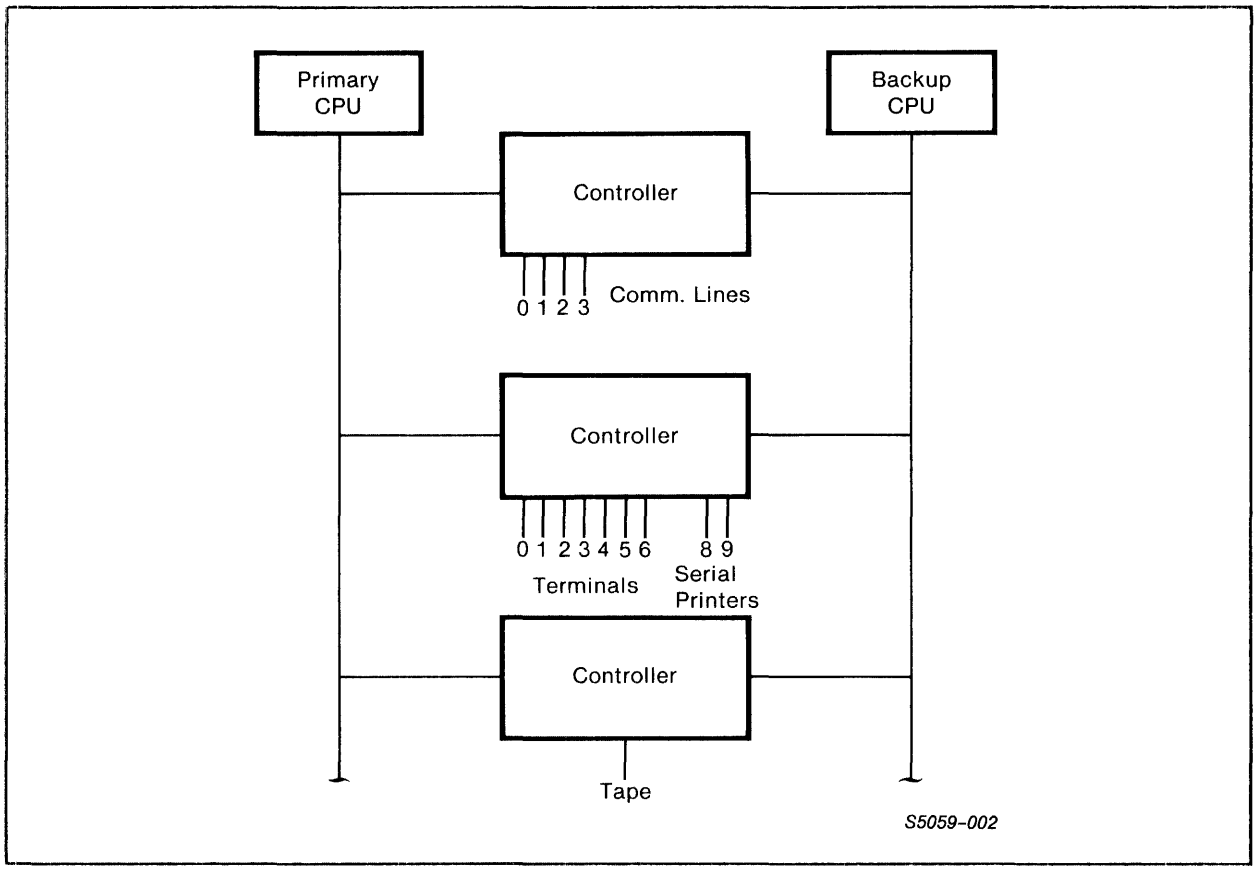


Figure 5-1. Nondisc Device Configuration

For dual-port nonmirrored and mirrored disc volumes, the configuration interfaces each drive to two controllers; that is, each drive is a dual-port disc. Figure 5-2 shows both a nonmirrored and a mirrored disc volume in a dual-port configuration.

The following PUP LISTDEV listing corresponds to the configuration shown in Figure 5-2:

```
LDEV   NAME     STATE  PPID    PCU    BPID    BCU DMNT TYPE SUBTYPE RECSIZE  DPTYPE

  8  $DISC-P      *   00,005 %112   01,005 %112   Y     3 A    3      4096      DP2
     $DISC-B          00,005 %12    01,005 %12
     $DISC-M      *   00,005 %113   01,005 %113
     $DISC-MB         00,005 %13    01,005 %13
  9  $ACCT-P          00,005 %14    01,007 %14          3 A    3      4096      DP2
     $ACCT-B      *   00,005 %164   01,006 %164
```

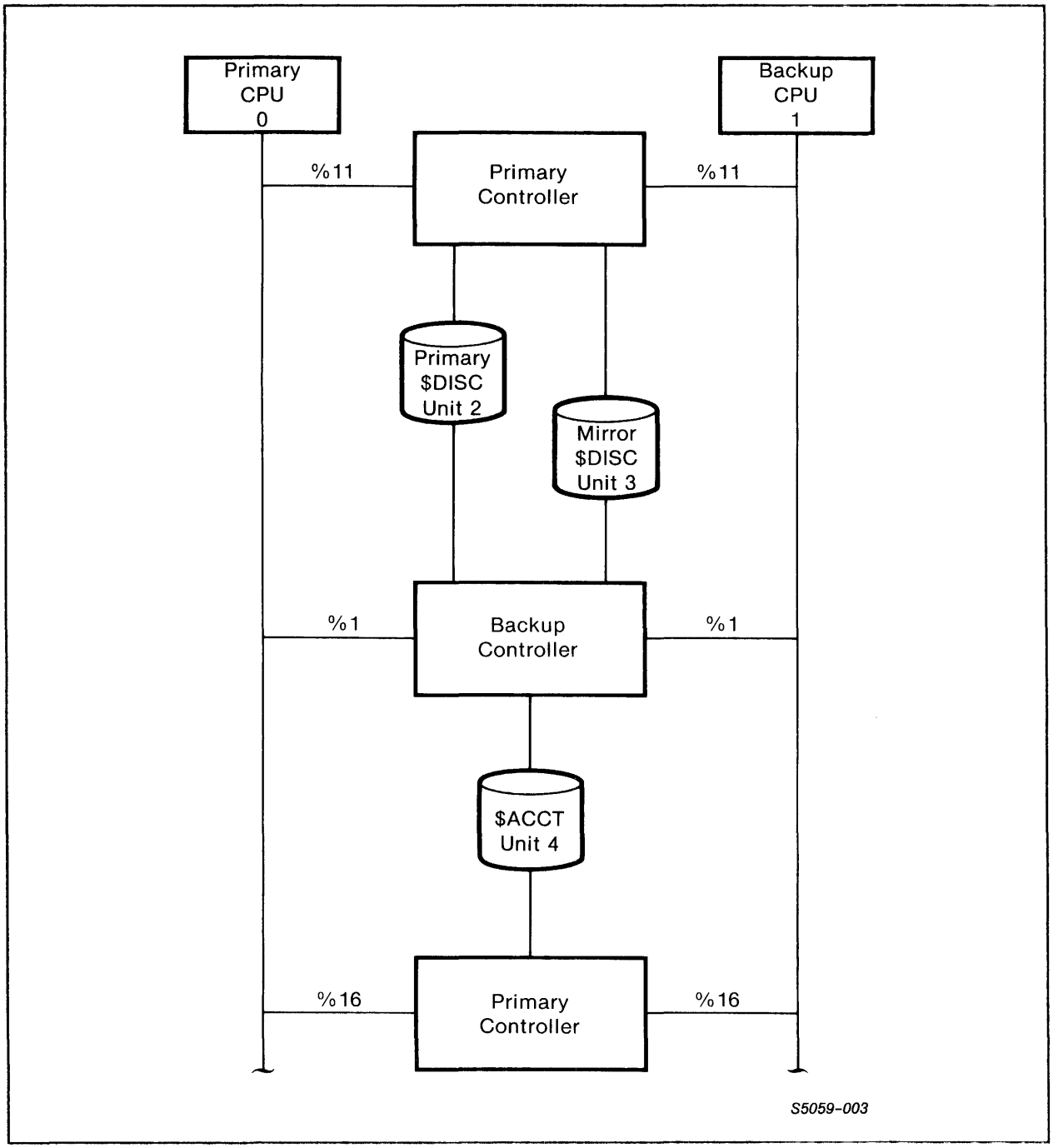


Figure 5-2. Nonmirrored and Mirrored Volumes in a Dual-Port Disc Configuration

Device References

The reference to a device can take one of four forms:

Form 1 Primary Path Designation

Form 2 Logical Device or Volume Designation

Form 3 Disc Device (Drive) Designation

Form 4 Controller Path Designation

These four designations are described on the following pages. Note that when applied to mirrored discs, the term device sometimes refers to one half of a mirrored disc pair and sometimes refers to the mirrored pair as a unit, depending on the context.

Form 1: Primary Path Designation

The primary path for a volume or device is designated at system generation. The primary path designation simply specifies which processor module (CPU) is to have primary control of the volume or device. (See Figure 5-3.)

The designation can be altered by the PRIMARY command or by hardware upon detection of an error. All communication with a device occurs through its primary path. (This includes both devices of a mirrored volume.)

The following PUP LISTDEV listing corresponds to the configuration shown in Figure 5-3. The asterisks indicate the primary paths.

```
LDEV   NAME     STATE  PPID    PCU    BPID    BCU DMNT TYPE SUBTYPE RECSIZE

  8  $DISC-P      *   00,005 %112   01,005 %112   Y    3 A    3       4096
     $DISC-B          00,005 %12    01,005 %12
     $DISC-M      *   00,005 %113   01,005 %113
     $DISC-MB         00,005 %13    01,005 %13
 38  $PRINT           00,007 %41    01,007 %41         5      1        132
 41  $TERM0           00,006 %42    01,006 %42         6      2         80
 46  $TERM4           00,006 %43    01,006 %43         6      4         80
```

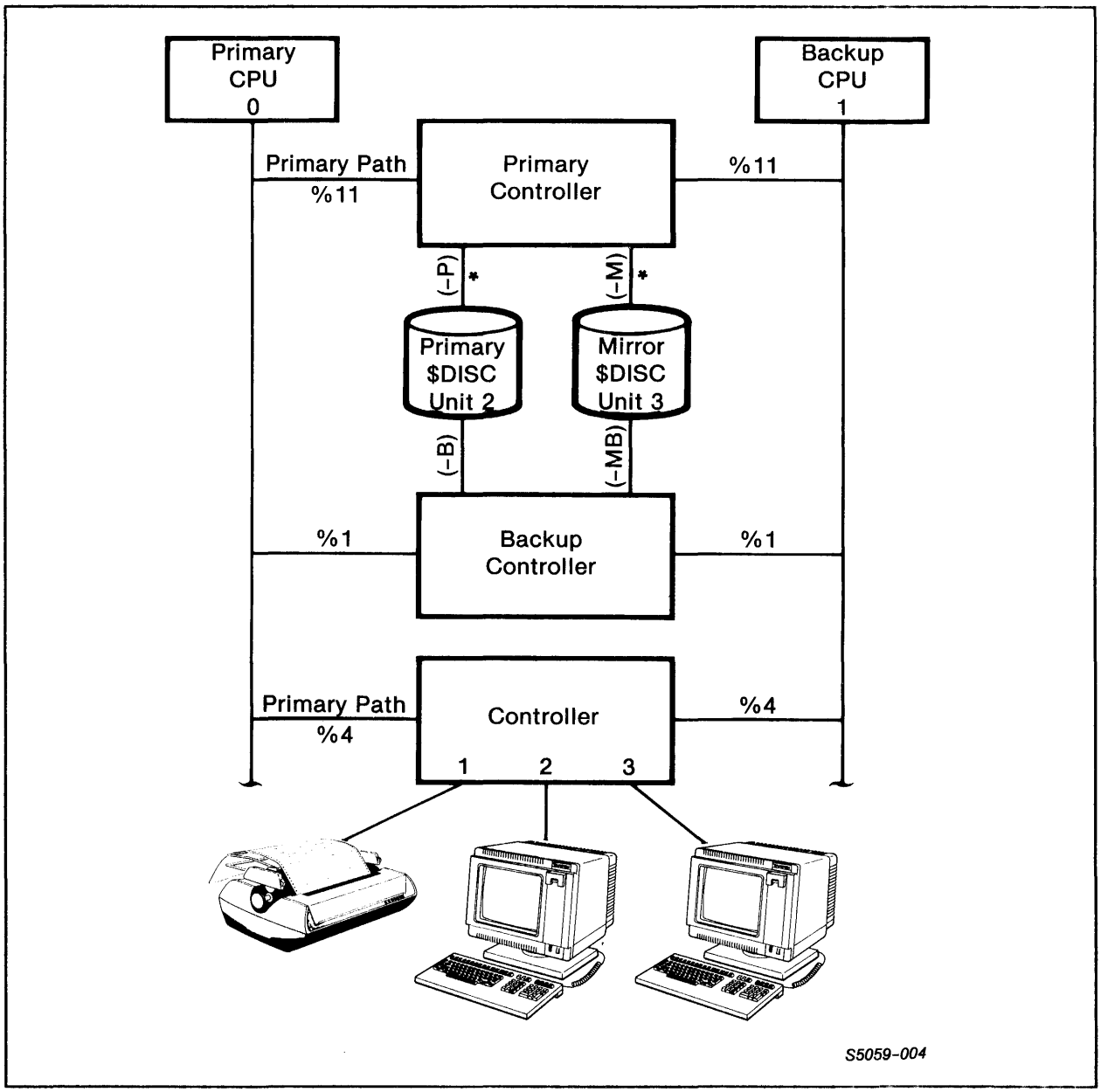


Figure 5-3. Primary Path Designation

Form 2: Logical Device or Volume Designation

A logical device or volume is specified in one of the following forms:

- $<volume-name> (refers to a disc device)
- $<device-name> (refers to any device)
- $<logical-device-number> or $<ldev-number> (refers to any device)

The following PUP LISTDEV listing corresponds to the configuration shown in Figure 5-4:

```
LDEV   NAME     STATE  PPID    PCU    BPID    BCU DMNT TYPE SUBTYPE RECSIZE

  8  $DISC-P      *  00,005 %112   01,005 %112   Y    3 A   3       4096
     $DISC-B         00,005 %12    01,005 %12
     $DISC-M      *  00,005 %113   01,005 %113
     $DISC-MB        00,005 %13    01,005 %13
 38  $PRINT          00,007 %41    01,007 %41         5     2        132
 41  $TERM0          00,006 %42    01,006 %42         6     2         80
 46  $TERM4          00,006 %43    01,006 %43         6     4         80
```

The following PUP commands refer to volume names or logical device numbers or call for them as parameters:

ADDRTOCYL (nonmirrored volume)
ADDRTOFILE
ALLOWOPENS
CHECKSUM
CONSOLE
COPY
CYLTOADDR (nonmirrored volume)
DEMOUNT (device form)
DOWN (device form)
FIXMICROCODE (volume name)
FORMAT (nonmirrored volume)
LABEL
LISTCACHE
LISTDEV
LISTFREE
PRIMARY (device form)
REBUILDDFS
REFRESH
RENAME
REPLACEBOOT
REVIVE
SETCACHE
STOPOPENS
UP (device form)

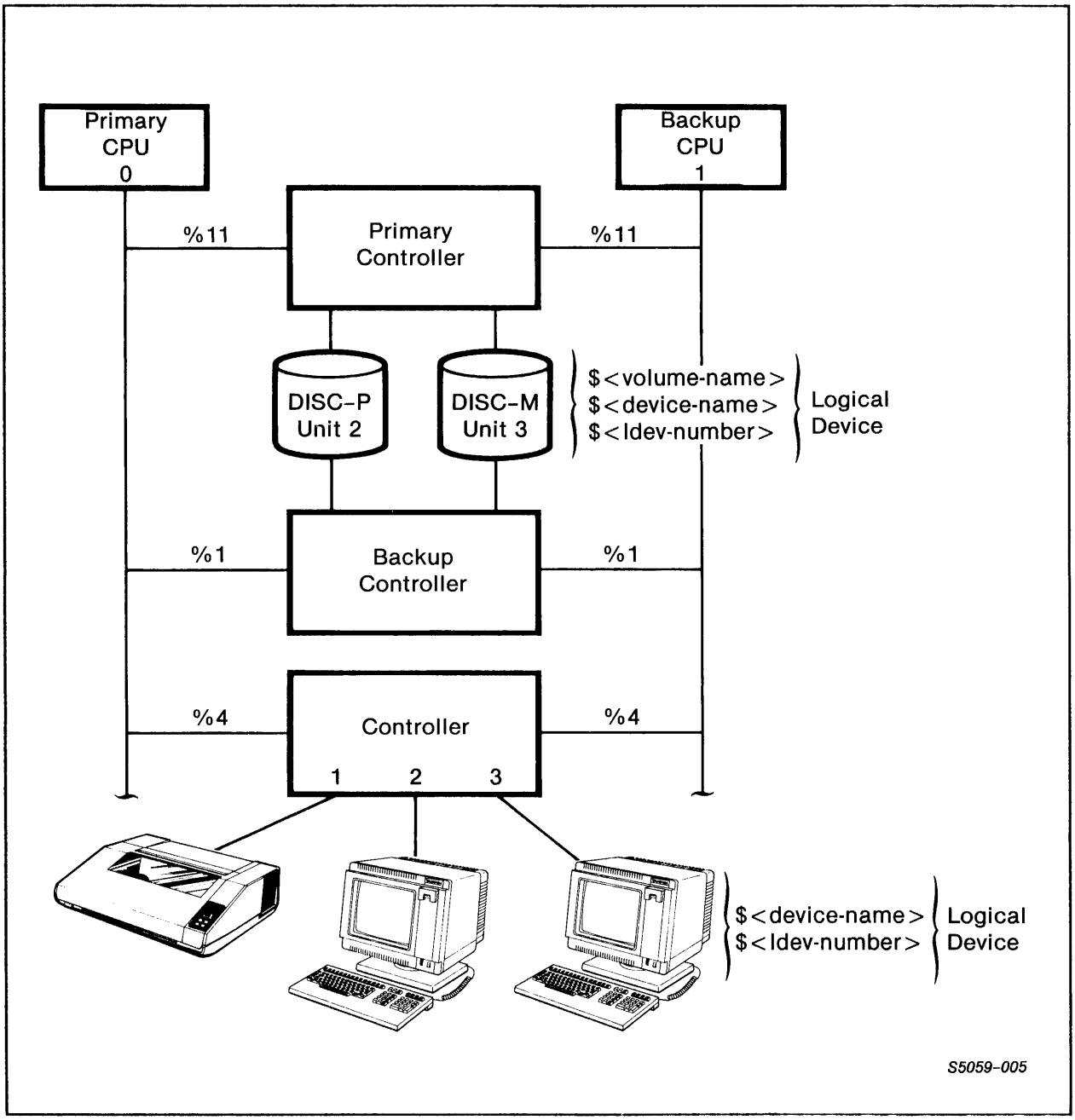


Figure 5-4. Logical Device or Volume Name Designation

Form 3: Disc Device (Drive) Designation

A disc drive is specified in one of the following forms:

$<volume-name> [-P / -M] or $<ldev-number> [-P / -M]

The following PUP LISTDEV listing corresponds to the configuration shown in Figure 5-5:

```
LDEV   NAME     STATE  PPID    PCU    BPID    BCU  DMNT TYPE SUBTYPE RECSIZE  DPTYPE
  8  $DISC-P      *   00,005  %112   01,005  %112   Y     3 A    3      4096     DP2
     $DISC-B          00,005  %12    01,005  %12
     $DISC-M      *   00,005  %113   01,005  %113
     $DISC-MB         00,005  %13    01,005  %13
  9  $ACCT-P          00,005  %14    01,005  %14    Y     3 A    3      4096     DP1
     $ACCT-M      *   00,005  %164   01,005  %164
```

The following PUP commands refer to disc drives or require them as parameters:

ADDRTOCYL	LISTBAD	LISTLOG	REMOVEAUDITED
CYLTOADDR	LISTDEFECTS	LISTSPARES	SPARE
FORMAT	LISTHEADERS	REMOVE	

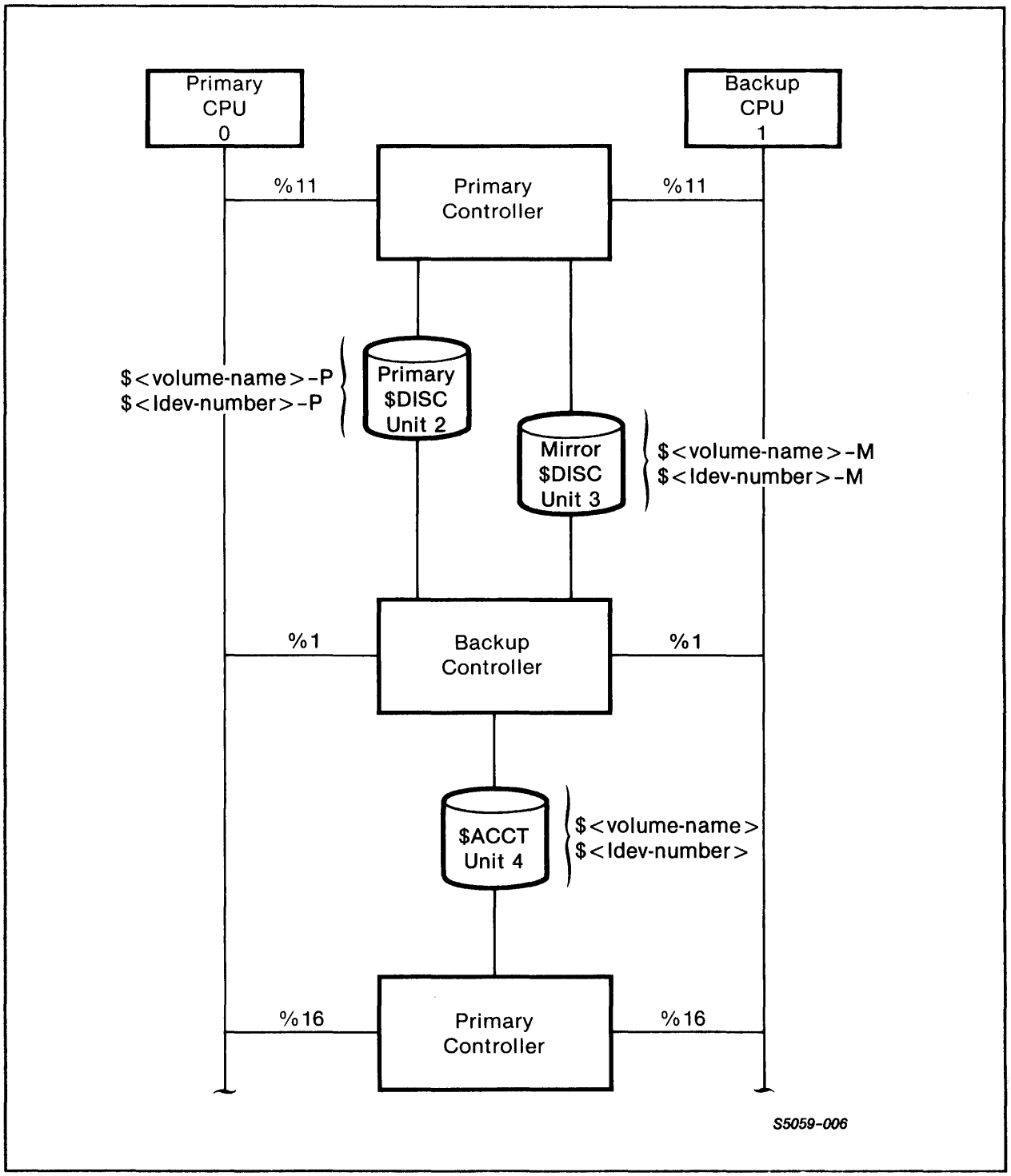


Figure 5-5. Disc Device Designation

Form 4: Controller Path Designation

A path to a disc device is specified in one of the following forms:

$<volume-name> [-P | -M | -B | -MB] or $<ldev-number> [-P | -M | -B | -MB]

The controller path designation does not alter the primary CPU designation. See Figure 5-6 for an illustration of these naming conventions. The following PUP LISTDEV listing corresponds to the configuration shown in Figure 5-6:

```
LDEV   NAME     STATE  PPID     PCU    BPID     BCU  DMNT TYPE SUBTYPE RECSIZE  DPTYPE

  8  $DISC-P      *   00,005  %112   01,005  %112   Y     3 A    3       4096     DP2
     $DISC-B          00,005  %12    01,005  %12
     $DISC-M      *   00,005  %113   01,005  %113
     $DISC-MB         00,005  %13    01,005  %13
```

The following PUP commands refer to controller paths or require them as parameters:

PRIMARY (preferred path form)
DOWN
UP

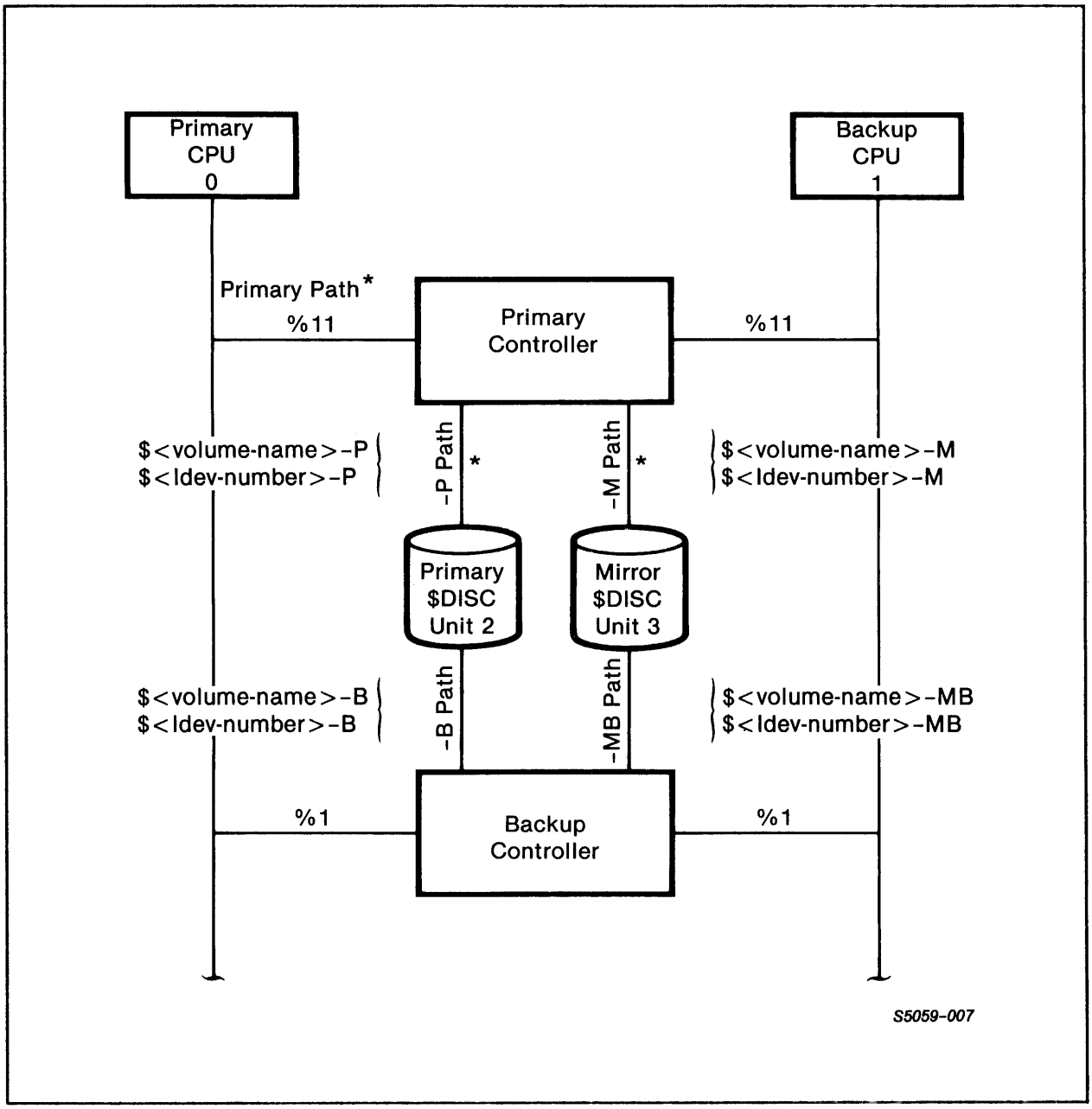


Figure 5-6. Controller Path Designation

PUP COMMAND SUMMARY

PUP commands fall into four categories: control commands, the command for the operator console, commands that apply to any device, and commands that apply only to disc devices. These command functions, grouped by category, are summarized in Table 5-1. The "User" column in Table 5-1 has two different symbols that represent:

Stan = Standard user
SG = Super-group user only (see the GUARDIAN Operating System User's Guide for additional information about user IDs).

Later in this section, the individual PUP commands are listed, in alphabetical order, and described in detail. PUP error messages appear in the System Messages Manual.

Table 5-1. PUP Command Summary (Continued on next page)

PUP Control Commands

Command Name	User	Description of Command
BREAK key	Stan	Aborts the currently executing "listing" type command. If a command is not being executed when the BREAK key is pressed, control returns to the command interpreter. Type PAUSE to return to PUP.
COMMENT	Stan	Causes PUP to ignore the rest of the line on which this command is found.
EXIT	Stan	Terminates PUP.
FC	Stan	(Fix command) is used to edit or cause re-execution of a command line. See the command interpreter FC command discussion elsewhere in this manual for an explanation.
HELP	Stan	Displays all PUP command names or the syntax of a particular command.

Table 5-1. PUP Command Summary (Continued)

PUP Command for the Operator Console

Command Name	User	Description of Command
CONSOLE	SG	Specifies the console device and the disc log file, and determines which messages are logged to the disc log file and the console listing device.

PUP Commands for Any Device

Command Name	User	Description of Command
DOWN	SG	Makes a device inaccessible to user processes.
FIXMICROCODE	SG	Rebuilds controller microcode section of the system disc.
LISTDEV	Stan	Lists configuration characteristics of devices on the system.
LOADMICROCODE	SG	Downloads the microcode file to a downloadable controller if all devices connected to the controller are in the down state.
PRIMARY	SG	Makes a designated processor module the primary path for a specified device, or designates a preferred controller path for a dual-port disc.
STATUS	Stan	Displays identification information of a downloadable controller. For the 3206 Tri-Density Tape Controller, this command display its idle loop and failure history.
UP	SG	Makes a down device accessible to user processes.
UPDATEREV	SG	Intended for use by Tandem customer engineers.

Table 5-1. PUP Command Summary (Continued)

PUP Commands for Disc Devices

Command Name	User	Disc Subtype	Description
ADDRTOCYL	Stan	all	Converts a logical or physical disc address for a specified volume into the corresponding cylinder, head, and sector.
ADDRTOFILE	Stan	all	Converts a logical disc address to a file name and offset within the file.
ALLOWOPENS	SG	all	Permits files to be opened on the specified volume.
CHECKSUM	SG	all	Recomputes the checksum value for specified portions of a disc volume.
COPY	SG	all	Creates a backup copy of one volume on another volume.
CYLTOADDR	Stan	all	Converts a cylinder, head, and sector for a specified volume into a corresponding logical or physical address.
DEMOUNT	SG	all	Clears the name of a down, demountable disc from the system so that the volume can later be mounted on a different spindle.
FORMAT	SG	all	Initializes sectors and creates a sector information table when a new, uninitialized pack is introduced into the system.
LABEL	SG	all	Writes a volume label on a disc pack that has previously been formatted (with the FORMAT command). Optionally, LABEL then transfers operating system images and files from a system image tape onto the volume. LABEL also allows you to set the cache configuration for the volume at volume-creation time.
LISTBAD	Stan	all	Lists the disc sectors marked as bad (but not yet spared) and the associated files on the specified disc volume.
LISTCACHE	Stan	all	Allows you to examine the current cache configuration for a volume (DP2 discs only).
LISTDEFECTS	Stan	all	Scans an entire disc or a cylinder or head of a disc for sectors whose headers have been flagged as defective by a FORMAT or SPARE operation.
LISTFREE	Stan	all	Lists the number of free disc pages, the size of contiguous blocks of free pages, the number of files, and an estimated maximum number of files on the specified volume.
LISTHEADERS	Stan	all	Lists the sector headers for an entire disc or a specified head or cylinder.
LISTLOG	Stan	all	Prints the permanent log of defective sectors.
LISTSPARES	Stan	all	Lists the assigned cylinder extensions and associated files on the specified volume.

Table 5-1. PUP Command Summary (Concluded)

PUP Commands for Disc Devices (Concluded)

Command Name	User	Disc Subtype	Description
REBUILDDFS	SG	all	Rebuilds the disc free space table.
REFRESH	Stan	all	Writes current file control block information to file labels on the disc volume, ensuring that these file labels represent the actual state of the files.
REMOVE	SG	all	Performs an implicit REFRESH, ALL and DOWN operation on a volume of a mirrored pair of discs so that the disc can be physically removed.
REMOVEAUDITED	SG	all	Enters in the Transaction Monitoring Facility (TMF) catalog the volume name and the names of all the audited files on a volume to be removed (for DP1 discs only).
RENAME	SG	all	Changes the name of a demountable disc volume without destroying the data on the volume.
REPLACEBOOT	SG	all	Allows the replacement of the microcode and disc cold-load bootstraps without performing a tape-to-disc cold load.
REVIVE	SG	all	Brings a down device of a mirrored pair of discs back into operation by copying data from the operable device to the downed device.
SETCACHE	SG	all	Alters the cache configuration for an in-use volume (DP2 discs only).
SPARE	SG	all	Assigns an alternate sector (from one of the spare sectors on the specified volume) to a sector indicated as bad by the LISTBAD command.
STOPOPENS	SG	all	Disallows any subsequent open operations for files on the specified volume.

PUP COMMAND SYNTAX

ADDRTOCYL Command

The ADDRTOCYL command converts a logical or physical disc address for a specified volume into the corresponding cylinder, head, and sector; it sends the result to the OUT <filename> (the default is the home terminal).

The syntax of the ADDRTOCYL command is:

```
ADDRTOCYL { $<volume-name> [-P] }
          { $<ldev-number> [-M] }, [%] <ms-word-of-address>

          , [%] <ls-word-of-address>  [ , PHYS ]
```

$<volume-name>

is the name of the disc volume whose disc address is to be converted.

$<ldev-number>

is the logical device number of the volume whose disc address is to be converted.

-P | -M

specifies the primary or mirror device. When you configure a mirrored pair from drives of different subtypes, you must specify which device you want ADDRTOCYL information for. This is because address conversions are different for each drive type.

→

<ms-word-of-address>

is the most significant word of the logical byte address or the physical byte address to be converted. If you give the physical byte address, you must specify the PHYS option.

<ls-word-of-address>

is the least significant word of the logical byte address or the physical byte address to be converted. If you give the physical byte address, you must specify the PHYS option.

PHYS

indicates that the specified address is a physical byte address. If omitted, the address is a logical byte address.

Considerations

- Disc addresses can be represented in either of two ways:

  --Logical byte addresses, which are relative to the beginning of the disc

  --Physical byte addresses, which are also relative to the beginning of the disc but which include those sectors that can be accessed only by PUP for sparing sectors

- If the <ms-word-of-address> and <ls-word-of-address> entries are preceded by a percent sign (%), the entries are treated as octal values; otherwise, the entries are treated as decimal values. These entries are in the same format as the disc addresses displayed by console messages 04 and 05 for logical byte addresses and console messages 97 and 98 for physical byte addresses. (Refer to "Console Messages" in the System Messages Manual.)

- With a mirrored pair, if the primary and mirror drive types are different (for example, a 4104 drive with a 4114 or 4115 drive), you must specify the volume (-P or -M). This is because the address conversions are different for each drive type.

Listing Format

The format of the ADDRTOCYL display listing is shown in Figure 5-7 below.

```
          CYL    HEAD    SECTOR       BYTE ADDR
                                       MSW    LSW

            n      nn        nn    %nnnnnn.nnnnnn
```

Figure 5-7. ADDRTOCYL Listing Format

Listing headers are identified as follows:

CYL HEAD	together define a track on $<volume-name>.
SECTOR	is a sector within the track on $<volume-name>.
BYTE ADDR	is the byte address of the sector.

Examples

1. To display on your OUT listing device the cylinder, head, and sector numbers whose logical byte address is %000172.126000 of the volume $SYSTEM, use this command to produce the following display:

```
:PUP ADDRTOCYL $SYSTEM, %000172, %126000

CYL    HEAD    SECTOR      BYTE ADDR
                             MSW     LSW

106      0       14      %000172.126000
```

2. If your primary is a 4114 or 4115 Winchester-type disc (disc subtype 9) and your backup is a 4104 (disc subtype 3), and both halves are up, you must specify primary or backup when using ADDRTOCYL:

```
:PUP ADDRTOCYL $DATA-M, %003471, %143000
```

ADDRTOFILE Command

The ADDRTOFILE command converts a logical disc address to a file name and offset within the file; it sends the result to the OUT <filename> (the default is the home terminal).

The syntax of the ADDRTOFILE command is:

```
ADDRTOFILE { $<volume-name> } , [%] <ms-word-of-address>
           { $<ldev-number> }

          , [%] <ls-word-of-address>
```

$<volume-name>

is the name of the volume whose disc address is to be converted.

$<ldev-number>

is the logical device number of the volume whose disc address is to be converted.

<ms-word-of-address>

is the most significant word of the logical byte address to be converted.

<ls-word-of-address>

is the least significant word of the logical byte address to be converted.

Considerations

- If the <ms-word-of-address> and <ls-word-of-address> entries are preceded by a percent sign (%), the entries are treated as octal values; otherwise, the entries are treated as decimal values. These entries are in the same format as the disc addresses displayed by console messages 04 and 05 for logical byte addresses. (Refer to "Console Messages" in the System Messages Manual.)
- If you specify a logical address that lies in free space or in space not designated for files, the ADDRTOFILE command displays nothing.
- If you specify a logical address that is located in multiple files, the ADDRTOFILE command lists the first file it finds.

Listing Format

The format of the ADDRTOFILE display listing is shown in Figure 5-8 below.

```
FILE NAME              FILE ADDRESS

<filename>         <relative-byte-address>
```

Figure 5-8. ADDRTOFILE Listing Format

Listing headers are identified as follows:

FILE NAME	is the file associated with the address on $<volume-name>.
FILE ADDRESS	is the <relative-byte-address> in <filename> associated with the address on $<volume-name>.

Example

On the volume named $DATA, to get the file name and file address of a file that corresponds to logical byte address %000712.126007, use this command to produce the following display:

```
:PUP ADDRTOFILE $DATA, %000712, %126007

     FILE NAME              FILE ADDRESS

 T9929A00.ESPDRSRC              201735
```

Note that this command can take several minutes to complete.

ALLOWOPENS Command

The ALLOWOPENS command permits files to be opened on the specified disc volume.

The syntax of the ALLOWOPENS command is:

```
ALLOWOPENS $<volume-name>  [ , SUPERONLY ]
```

$<volume-name>

is the name of the volume where file-open operations are allowed.

SUPERONLY

causes the volume to enter a state in which only the super ID user (with a user ID of 255,255) can open a file on that volume. If omitted, any user can open a file on that volume.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- The PUP STOPOPENS command (which refuses file-open requests) is cleared by the ALLOWOPENS command.
- It is not necessary to execute a STOPOPENS command if you issue an ALLOWOPENS command with the SUPERONLY option.
- To clear the SUPERONLY condition, enter an ALLOWOPENS command that does not specify SUPERONLY, or enter a STOPOPENS command for the disc volume.

Examples

1. To allow open requests on the volume named $BOOKS1, enter:

```
:PUP ALLOWOPENS $BOOKS1
```

2. To allow open requests by only the super ID user (user ID 255,255) on the volume named $DATA, enter:

```
:PUP ALLOWOPENS $DATA, SUPERONLY
```

CHECKSUM Command

The CHECKSUM command recomputes the checksum value for specified portions of a disc volume.

NOTE

Typically, the CHECKSUM command is issued only by a Tandem representative when recovering from a software-detected checksum error.

The syntax of the CHECKSUM command is:

```
CHECKSUM  { $<volume-name> }  , [%] <ms-word-of-address>
          { $<ldev-number> }

          , [%] <ls-word-of-address>  [ , <byte-length> ]
```

$<volume-name>

is the name of the volume containing a portion whose checksum is to be calculated.

$<ldev-number>

is the logical device number of the volume containing a portion whose checksum is to be calculated.

<ms-word-of-address>

is the most significant word of the beginning logical byte address of the portion of the disc whose checksum is to be calculated.

→

<ls-word-of-address>

is the least significant word of the beginning logical byte address of the portion of the disc whose checksum is to be calculated.

<byte-length>

is the number of bytes in the portion of the disc whose checksum is to be calculated. If <byte-length> is omitted, a length of 512 is assumed.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- The CHECKSUM command does not work if the disc is not in a down state or if files are open on the volume.
- The File Utility Program (FUP) also has a CHECKSUM command (described in Section 3 of this manual). PUP CHECKSUM and FUP CHECKSUM are used in different situations.
- PUP CHECKSUM corrects only sector checksums for some range of disc addresses, whether or not the address is within a file.
- FUP CHECKSUM corrects sector and block checksums on structured files and sector checksums on unstructured files. If the checksum error is within a file (determined by using the PUP ADDRTOFILE command), then use FUP CHECKSUM to correct it. If the checksum error is in free space or in a space not designated for files, use PUP CHECKSUM.
- See the System Operator's Guide for additional information on the procedure "Recovering from a Checksum Error," which discusses the contexts that require FUP CHECKSUM rather than PUP CHECKSUM.

- A sector with a bad checksum is rewritten with a good checksum. However, if the initial checksum error was due to bad data, the data itself remains bad even though the checksum recomputed on it is now good. In this case, you no longer encounter checksum errors, but you still have bad data. Because a checksum error may indicate a hardware or system software problem, you should report the error to your Tandem representative.

- Console messages that indicate a software-detected checksum error contain parameters for the <ms-word-of-address> and <ls-word-of-address>. If the <ms-word-of-address> and <ls-word-of-address> entries are preceded by a percent sign (%), the entries are treated as octal values; otherwise the entries are treated as decimal values. These entries are in the same format as the disc addresses displayed by console messages 04 and 05 for logical byte addresses. (Refer to "Console Messages" in the System Messages Manual.)

- When the CHECKSUM operation completes, the disc is automatically put in the up state. No REVIVE operation is performed.

- The following procedure shows the steps necessary to bring the system disc down so that a PUP CHECKSUM can be performed:

Correcting a Checksum Error on $SYSTEM

CAUTION

This procedure should be viewed as an emergency action. We do not guarantee that it will always work even if the system is quiesced. If this procedure fails, it may leave $SYSTEM in an unusable state. To prepare for recovery, you should first remove the $SYSTEM mirror or be prepared to do a cold load from tape (this destroys all existing files on the $SYSTEM disc).

NOTE

After performing this procedure, you must cold load the system.

```
1. :LOGON SUPER.SUPER
2. :FUP DUP $SYSTEM.SYS<nn>.COMINT, $HOLD.TEST.COMINT
3. :FUP DUP $SYSTEM.SYS<nn>.PUP, $HOLD.TEST.PUP
4. :RUN $HOLD.TEST.COMINT / IN $<term>, OUT $<term>, NOWAIT/
5. :EXIT
```

Steps 2 and 3 copy the COMINT and PUP program files to another volume, $HOLD, and Step 4 starts another command interpreter there. Step 5 exits you from the command interpreter running on $SYSTEM.

```
6. :LOGON SUPER.SUPER
```

At this point, you have logged onto the GUARDIAN command interpreter that is running on the volume $HOLD. Stop any applications that are running on $SYSTEM and notify all users to close any open files (use FUP LISTOPENS to find out what files are still open). All activity must be stopped before proceeding to the next step.

```
 7. :RUN $HOLD.TEST.PUP
 8. :PUP
 9. #DOWN all line handlers
10. #REFRESH $SYSTEM, ALL
11. #STOPOPENS $SYSTEM
12. #DOWN ! $SYSTEM
13. #CHECKSUM $SYSTEM, %000172, %001100
```

Example

To calculate a checksum of 512 bytes in the portion of disc beginning with the word address %000172.126007 on the volume named $DATA, enter:

```
:PUP
#CHECKSUM $DATA, %000172, %126007
```

COMMENT Command

The COMMENT command allows you to include lines of explanation or documentation in your OBEY files. When you execute the file, the command interpreter ignores any lines that are prefixed by the word COMMENT.

The syntax of the COMMENT command is:

```
COMMENT  <text>

<text>

   is a line of text.
```

Example

This example of an OBEY file contains a PUP command. When the file executes, the command interpreter ignores the lines in the file that begin with the word COMMENT.

```
:PUP
COMMENT  This is a comment.
COMMENT  This file lists configuration information for all
COMMENT  type 3 devices and prints the listing on device $LP.
#LISTDEV /OUT $LP/TYPE 3
#EXIT
```

CONSOLE Command

The CONSOLE command can perform two different kinds of operations:

1. It can redesignate the device that is to serve as the operator console hard-copy device. This device can be either on the local system or on a remote system.

2. It can perform one of the following operations:

   --Switch disc log files

   --Turn on and off logging of system messages or user messages to the disc log file

   --Enable and disable logging of system messages or user messages on the console hard-copy device

The syntax of the CONSOLE command is:

```
CONSOLE [ [ \<system-name>.]$<device-name>  ]
        [ [ \<system-name>.]$<ldev-number>  ]
        [ [ \<system-name>.]$0              ]

        [ , ON | , OFF | , SWITCH                          ]
        [ , ENABLE  { <number> | * }                        ]
        [ , DISABLE { <number> | * }                        ]
        [ , SHOW    { <number> | * }                        ]
        [ , USER    { <user-id> | <user-name> }             ]
        [ , USERMSG { ON | OFF }                            ]
```

\<system-name>

is the remote system in which the device specified is located. When specifying a device on a remote system, the USER parameter, below, is required. If omitted, the device is assumed to be on the local system.

$<device-name>

is the name of the designated device for operator message logging. This device cannot be a disc or tape unit.

→

$<ldev-number>

is the logical device number of the designated device for console message logging. This device cannot be a disc or tape unit.

$0

turns off logging of console messages to the hard-copy device.

ON

turns on logging of console messages to the disc log file and the hard-copy device.

OFF

turns off logging of console messages to the hard-copy device.

SWITCH

directs the operator process to close, then reopen, the disc log file. If the proper three-step procedure has been performed before executing the switch, this option directs console messages to a new log file. (The three-step procedure is described below in example 4.)

ENABLE { <number> | * }

resumes logging to the disc log file and hard-copy device of a specific message <number> or of all console messages (indicated by *). That is, either a specified console message or all console messages are logged. If <number> is zero, user messages are enabled. If you specify the ENABLE option, PUP responds with a message stating that either all messages or a specific message will appear.

→

DISABLE { <number> | * }

suppresses logging to the disc log file and hard-copy device of of a specific message <number> or of all console messages (indicated by *). That is, either a specific message or all messages are not logged. If <number> is zero, user messages are disabled. If you specify the DISABLE option, PUP responds with a message stating that either a specific message or all messages will appear.

SHOW { <number> | * }

displays the state (either enabled or disabled) of all console messages (indicated by *) or of the specific message <number>.

USER { <user-id> | <user-name> }

specifies a local user with access to the remote system to which console logging is being directed. This parameter is used only when a remote logging device is specified. Either the user ID or the user name can be specified. The default user is NULL.NULL.

USERMSG { ON | OFF }

turns on and off the logging of user messages to the disc log file. The default is USERMSG ON.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for information about user IDs.)
- Console messages consist of both system-generated and user-generated messages. The various kinds of system console messages are described in the System Messages Manual. User messages are messages produced by user-written applications; they can appear on either the hard-copy device (often called $CONSOL) or a user-written application process (usually named $AOPR), as well as the disc log file (the OPRLOG).

- If a device on a remote system is designated the console logging device, the user specified in the USER parameter must have access to the remote system. To have access to a remote system, a user must establish remote passwords with that system. (Refer to the COMINT command REMOTEPASSWORD in Section 2 of this manual for information about this procedure.)

- Table 5-2 shows how the various CONSOLE options can affect the logging of console messages to the disc log file and the hard-copy device.

Table 5-2. How PUP CONSOLE Affects Message Logging

CONSOLE Option	System Messages	User Messages	Hard-Copy Device	Disc Log File
$0	X	X	Off	On
ON*	X	X	On	On
OFF	X	X	On	Off
ENABLE*	X	X	On	On
DISABLE	X	X	Off	Off
USERMSG ON*		X	On	On
USERMSG OFF		X	On	Off

* = Default

- To determine the logging status of a disc, execute a PUP CONSOLE, SWITCH command. The responding message gives the disc log file in "LOG TERMINAL LDEV # <nnnnn>" and the status of logging to the hard-copy device in "DISC LOGGING ON|OFF".

- For users without super-group status, a PUP LISTDEV $0 command gives the disc log file under the "STATE" header and the status of logging to the hard-copy device under the "PCU" header.

- If a device on a remote system is designated the console logging device, the user specified in the USER parameter must have access to the remote system. To have access to a remote system, a user must establish remote passwords with that system. (Refer to the COMINT command REMOTEPASSWORD in Section 2 of this manual for information about this procedure.)
- Table 5-2 shows how the various CONSOLE options can affect the logging of console messages to the disc log file and the hard-copy device.

Table 5-2. How PUP CONSOLE Affects Message Logging

CONSOLE Option	System Messages	User Messages	Hard-Copy Device	Disc Log File
$0	X	X	Off	On
ON*	X	X	On	On
OFF	X	X	On	Off
ENABLE*	X	X	On	On
DISABLE	X	X	Off	Off
USERMSG ON*		X	On	On
USERMSG OFF		X	On	Off

* = Default

- To determine the logging status of a disc, execute a PUP CONSOLE, SWITCH command. The responding message gives the disc log file in "LOG TERMINAL LDEV # <nnnnn>" and the status of logging to the hard-copy device in "DISC LOGGING ON|OFF".
- For users without super-group status, a PUP LISTDEV $0 command gives the disc log file under the "STATE" header and the status of logging to the hard-copy device under the "PCU" header.

Examples

1. To log system console messages on printer $LP, enter:

```
:PUP CONSOLE $LP
```

2. To turn off logging to the hard-copy device, enter:

```
:PUP CONSOLE $0
```

3. To turn off logging of system console messages to the disc log file, enter:

```
:PUP CONSOLE, OFF
```

4. To turn off logging of system console messages to both the disc log file and the printer, enter:

```
:PUP CONSOLE, DISABLE
```

5. To switch to a new log file, first rename the current disc log file by entering:

```
:VOLUME $SYSTEM.SYSTEM
:RENAME OPRLOG, LOGFILE
```

   Console message logging continues to LOGFILE. Now create a new log file with the File Utility Program (FUP) by entering:

```
:FUP CREATE OPRLOG, LIKE LOGFILE
```

   Finally, switch to the new log file by entering:

```
:PUP CONSOLE, SWITCH
```

   This command closes the current log file, opens the new log file OPRLOG, and performs a positioning operation. Subsequent system console messages are appended to the end of the new OPRLOG.

6. To designate the terminal $TERM4 on the remote system \OMNI as the console listing device, enter:

```
:PUP CONSOLE \OMNI.$TERM4, USER SUPER.JIM
```

   The user SUPER.JIM must have remote passwords established with system \OMNI.

COPY Command

The COPY command creates a backup copy of one disc volume on another disc volume. Both volumes must be of the same subtype (use the LISTDEV command to verify), and both must be in the down state before the COPY is attempted.

The syntax of the COPY command is:

```
COPY [ / OUT <listfile> / ]  $<in-vol-name>
     , $<out-vol-name> [ , VERIFY ]
```

<listfile>

is the device on which any errors are to be listed.

$<in-vol-name>

is the name of the volume to be copied.

$<out-vol-name>

is the name of the volume onto which the disc image is to be copied.

VERIFY

indicates that the COPY command is to verify each write to the $<out-vol-name> by reading the data just written.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)

- The disc image of $<in-vol-name> is duplicated on the designated $<out-vol-name>.
- The COPY command writes to the OUT <listfile> any errors that occur during the COPY operation.
- The COPY command does not allow you to copy a disc with a DP1 format to a disc with DP2 format, or vice versa. If you do attempt a PUP COPY with unlike disc-process types, the following error messages appear:

  ```
  COPY REQUEST DENIED BECAUSE OF TYPE PROCESS MISMATCH
  SOURCE VOLUME NAME IS A { DP1 | DP2 } DISCPROCESS
  DESTINATION VOLUME NAME IS A { DP1 | DP2 } DISCPROCESS
  ```

- In order to copy to an unlike disc-process type, you must convert the data files. For more information, refer to the DP1-DP2 File Conversion Manual.

Listing Format

The format of the COPY display listing is shown in Figure 5-9.

```
        DEV                ERROR                    ADDRESS

  $<volume-name>     <error-msg>      <logical-byte-address>
         .                  .                        .
         .                  .                        .
```

Figure 5-9. COPY Listing Format

Listing elements are identified as follows:

$<volume-name>	is the volume where the error occurred.
<error-msg>	is an error that occurred during the COPY operation and is one of the following: READ DATA ERROR--an error occurred while reading $<in-vol-name>. Invalid data is written in the corresponding sector on the $<out-vol-name>, where sector length = 512 bytes.

READ DATA LOST--an error occurred while reading $<in-vol-name>. Zeros are written in the corresponding sector on $<out-vol-name>.

WRITE DATA LOST--an error occurred while writing to $<out-vol-name>. Data in the associated sector may be invalid.

<logical-byte-address> is the byte address of the sector where the error occurred.

Examples

1. To copy the entire disc image (including the system area, file labels, free space table, and directory) on the volume named $DBASE to the volume named $DBSBACK, enter:

```
:PUP COPY $DBASE, $DBSBACK
```

2. To copy the entire disc image on the volume named $OLD to the volume named $NEW, verify the data, and list the errors on the device named $LP, enter:

```
:PUP COPY /OUT $LP/ $OLD, $NEW, VERIFY
```

CYLTOADDR Command

The CYLTOADDR command converts a cylinder, head, and sector number for a specified disc volume into a corresponding logical or physical byte address (see ADDRTOCYL command). The result appears in the OUT <listfile> (the default is your home terminal).

The syntax of the CYLTOADDR command is:

```
CYLTOADDR { $<volume-name> [-P] }
          { $<ldev-number> [-M] }, <cylinder> , <head> ,

          <sector> [ , PHYS ]
```

$<volume-name>

is the name of the volume whose <cylinder>, <head>, and <sector> number are to be converted.

$<ldev-number>

is the logical device number of the volume whose <cylinder>, <head>, and <sector> number are to be converted.

-P | -M

specifies the primary or mirror device. When you configure a mirrored pair from drives of different types, you must specify which device you want CYLTOADDR information for. This is because address conversions are different for each drive type.

<cylinder>

is the cylinder number to be converted.

→

<head>

is the head number to be converted.

<sector>

is the sector number to be converted.

PHYS

indicates that the address to be produced is a physical byte address. If omitted, a logical byte address is produced.

Considerations

- The physical address takes into account those sectors reserved for sparing within each cylinder, as explained under the ADDRTOCYL command.
- With a mirrored pair, if the primary and mirror drive types are different (for example, a 4104 drive mirrored with a 4114 or a 4115), you must specify the volume (-P or -M). This is because the address conversions are different for each drive type.

Listing Format

The format of the CYLTOADDR display listing is the same as that for the ADDRTOCYL command.

Examples

1. On the volume named $SYSTEM, to convert cylinder number 27, head number 2, sector number 5 to a logical byte address, enter:

```
:PUP CYLTOADDR $SYSTEM, 27, 2, 5

 CYL    HEAD    SECTOR      BYTE ADDR
                              MSW    LSW

  27     2        5       %000037.135000
```

2. If your primary is a 4114 or 4115 Winchester-type disc (disc subtype 9) and your backup is a 4104 (disc subtype 3) and both halves are up, you must specify primary or backup when using CYLTOADDR:

```
:PUP CYLTOADDR $DATA-M, 13, 4, 8
```

DEMOUNT Command

The DEMOUNT command clears the name of a down, demountable disc from the system. The volume can then be mounted on a different spindle, or some other volume can be mounted.

The syntax of the DEMOUNT command is:

```
DEMOUNT { $<volume-name> }
        { $<ldev-number> }
```

$<volume-name>

is the name of the volume to be cleared from the system.

$<ldev-number>

is the logical device number of the volume to be cleared from the system.

Consideration

This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)

Example

To clear the volume named $BOOKS3 from the system, enter:

```
:PUP DEMOUNT $BOOKS3
```

DOWN Command

The DOWN command makes a device inaccessible to user processes.

The syntax of the DOWN command is:

```
DOWN [!] { $<device-name>                          }
         { $<ldev-number>   [-P]                   }
         {                  [-M]                   }  [ , HARD ]
         {                  [-B]                   }
         {                  [-MB]                  }
         { $<volume-name>   [-P]                   }
         {                  [-M]                   }
         {                  [-B]                   }
         {                  [-MB]                  }
```

!

means place the device down even if files are open on the device.

NOTE

Do not DOWN ! $SYSTEM unless ALL files have been closed and PUP and COMINT have previously been copied to another disc. To down the system disc, perform the procedure that is described under the CHECKSUM command in this section, omitting the CHECKSUM step.

$<device-name>

is the name of a device (nondisc) to be brought down.

$<ldev-number>

is the logical device number of the device to be brought down.

→

$<volume-name>

is the name of the nonmirrored or mirrored disc volume to be brought down.

NOTE

The path specifiers -P, -M, -B, and -MB apply only to dual-port discs.

-P

places the primary controller path to the disc device down. This path specifier can refer either to a nonmirrored device or to the primary half of a mirrored device.

-M

places the primary controller path to the mirror half of a mirrored device down.

-B

places the backup controller path to the disc device down. This path specifier can refer either to a nonmirrored device or to the primary half of a mirrored device.

-MB

places the backup controller path to the mirror half of a mirrored device down.

HARD

places a specified path to a disc device in the "hard down" state. This option requires a path specifier (-P, -M, -B, or -MB).

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)

- This command is usually entered if a device is known to be malfunctioning; if the device has been physically removed from the system, and the system has not been reconfigured; or if a diagnostic program is to be run for the device.

- When you enter the DOWN command, PUP responds with a warning message and a verification request, as follows:

```
WARNING:  The DOWN operation will cause $<device-name>
          to be INACCESSIBLE to user processes

Are you sure you want to DOWN $<device-name> [ Y/N ]?
```

  The device is brought down only if you respond with "Y" or "y" or any combination of upper and lower case letters of "YeS". In all other cases, PUP ignores the DOWN command and returns a prompt for your next command.

- If any files are open when the DOWN command is given, PUP responds with:

```
<number> FILES OPEN
```

- The hard down state is recorded as an "H" in the state field of the LISTDEV display (refer to the description of the LISTDEV command later in this section). Any attempt to access a path in a hard down state fails with GUARDIAN file-system error 66 (DEVICE HAS BEEN DOWNED BY OPERATOR). (Refer to the System Messages Manual for additional information about GUARDIAN file-system errors.)

- A PUP UP command does not bring up a path from the hard down state. To bring a controller path out of a hard down state, execute another PUP DOWN command for the device, omitting the HARD option and specifying the controller path with its identifying tag (-P, -M, -B, or -MB).

Examples

1. To make the printer named $LLP inaccessible to user processes, enter:

```
:PUP DOWN $LLP
```

2. To make the primary controller path to the mirror device named $DATA inaccessible to user processes, enter:

```
:PUP DOWN $DATA-M
```

3. To make logical device number $6 inaccessible to user processes, enter:

```
:PUP DOWN $6
```

EXIT Command

The EXIT command terminates the PUP program and returns control to the command interpreter.

The syntax of the EXIT command is:

```
EXIT
```

Example

The following example illustrates the PUP EXIT command:

```
:PUP
#DEMOUNT $ACCT
#EXIT
:
```

FC Command

FC, or Fix command, allows you to modify and resubmit the last command line entered. The FC subcommands are the same as those used for the GUARDIAN command interpreter FIX command, described in Section 2.

The syntax of the FC command is:

```
FC
```

Example

In the following example, you want to find out the current state of the $BOOKS disc but did not enter the name correctly, causing an error message to appear:

```
#LISTDEV $BOOK
NO SUCH DEVICE
#FC
#LISTDEV $BOOK
.               S
#LISTDEV $BOOKS
.

 LDEV     NAME     STATE  PPID    PCU     BPID    BCU DMNT TYPE SUBTYPE RECSIZE DPTYPE

   12  $BOOKS-P      *  06,009 %210   07,009 %210   Y     3 A   6       4096   DP1
       $BOOKS-B         06,009 %310   07,009 %310
       $BOOKS-M      *  06,009 %211   07,009 %211
       $BOOKS-MB        06,009 %311   07,009 %311
#
```

FIXMICROCODE Command

The FIXMICROCODE command enables rebuilding of the controller microcode section of the system disc if that part of the system disc has been reported as bad (that is, defective).

The syntax of the FIXMICROCODE command is:

```
FIXMICROCODE  $volume-name [-P]
                           [-M]
```

$<volume-name>

is the name of the volume where the rebuilding of the controller microcode section is to be made.

-P | -M

indicates that the primary or mirror half of the mirrored volume is to be selected.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- You should issue this command when console message 161 (REVIVE FAILED TO FIX UP DISC MICROCODE SECTION) has been received, indicating that the controller section of the microcode on the system disc is bad.

Example

To rebuild the controller microcode section of the mirror half of the system disc $SYSTEM, enter:

```
:PUP
#FIXMICROCODE $SYSTEM-M
```

FORMAT Command

The FORMAT command introduces a new, uninitialized disc pack into the system or to reformat an existing pack on the system. It initializes sectors and creates a sector information table (SIT).

Optionally, a progress report can be requested to get information about the FORMAT operation.

CAUTION

Execution of the FORMAT command destroys any existing files on the volume being formatted.

The syntax of the FORMAT command is:

```
FORMAT [ / [ IN <filename> ] [,] [ OUT <filename> ] / ] [!]
       { $<volume-name> [-P]  }
       {                [-M]  }
       {                      }
       { $<ldev-number> [-P]  }
       {                [-M]  }
       [ , NORETAIN [ , STATUS [ <status-interval> ] ] ]
```

IN <filename>

specifies a file containing a list of sectors to be marked unconditionally as defective. If a terminal is specified as the IN <filename>, PUP prompts for the addresses. See "Considerations" for the address format.

OUT <filename>

designates the file where any error messages (generated by the FORMAT) and the progress report, if requested (by the STATUS option), are to be listed. If you omit this option, FORMAT directs its output to the home terminal.

→

!

indicates that PUP is to proceed with the FORMAT operation even if files are open on the volume to be formatted.

$<volume-name>

is the name of the volume to be formatted. If the volume is demountable, you must specify $<ldev-number> instead.

$<ldev-number>

is the logical device number of the volume to be formatted. If the volume is demountable, you must specify $<ldev-number>.

-P | -M

specifies that the FORMAT command is to be executed on the primary or mirror device. You must specify -P or -M if the volume to be formatted is a mirrored volume.

NORETAIN

causes PUP to format and test all sectors (except those listed in an IN <filename>) even those previously marked as defective. If PUP finds a sector to be good (even though it may be marginal), it passes the FORMAT and will contain a valid header.

STATUS [<status-interval>]

requests periodic progress reports as the formatting proceeds. The integer range for <status-interval> is 1 through 60, specifying the number of minutes between progress reports. If <status-interval> is not specified, the progress report defaults to five-minute intervals.

→

<status-interval> is the minimum time between reports, not the precise time; after each cylinder is processed, a check is made to see if another progress report should be displayed.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- If a sector fails the FORMAT, it is flagged as defective, and a PUP LISTDEFECTS performed subsequently for that disc will show the defective sector.
- PUP FORMAT does not add entries to the lifetime defect log; it only flags sectors as defective on the disc. You must perform a PUP SPARE in order to enter a newly flagged sector into the defect log.
- The NORETAIN option allows PUP to format a disc whose lifetime defect log has been made unreadable for any reason (such as hardware problems, alignment, destroyed data, or a new medium). If there is no previous defect log or both copies of the previous version are unreadable, the NORETAIN option formats the defect log cylinder, preparing it for the creation of a new defect log.
- If you do not specify NORETAIN, PUP FORMAT checks for the defect log and must be able to read it or else the FORMAT terminates.
- If you do not specify NORETAIN, PUP FORMAT reflags as defective those sectors with a flag field of %351 or %377 (as shown by a PUP LISTDEFECTS).
- If you specify NORETAIN, PUP FORMAT returns to use any previously spared sectors that pass this FORMAT. To flag as defective all sectors listed in the lifetime defect log, do a PUP LISTLOG with an OUT <listfile> (located on another disc) and use this as the IN <listfile> to the FORMAT>

- The IN <filename> option allows defective-sector information to be factored into the FORMAT. This information is a list of sector addresses, one for each record, in the following format (not necessarily in ascending order):

  <pcyl> [,] <phead> [,] <psect>

  <pcyl> is the physical cylinder address.

  <phead> is the physical head address.

  <psect> is the physical sector address.

- A physical cylinder on each disc pack is dedicated as the defect log cylinder. This cylinder is intended to contain a historical list of defective sectors on a disc pack. The lifetime defect log is created at Tandem Computers; it includes any defects contained in the vendor defect list as well as any that Tandem detected during the initial formatting of the disc. The rest of the log consists of defective sectors that have been spared.

- The lifetime defect log is maintained in the next-to-last available cylinder for each specific disc type. Two copies are maintained--one starting at the beginning of the cylinder and the second starting halfway through the cylinder. When using the NORETAIN option, if either of the two copies of the defect log is readable, the defect log cylinder is left intact. If both are unreadable, the defect log cylinder is reformatted like any other cylinder, and a new defect log is created by the next SPARE command.

- PUP has no command that removes entries from the lifetime defect log. Once the defect log is updated, an entry cannot be deleted. Only your Tandem representative can delete or overwrite the contents of the defect log.

- After completing a PUP FORMAT, execute the PUP LISTDEFECTS command to list all currently flagged sectors. Compare this to the list of defective sectors in the defect log given by the PUP LISTLOG command. Using PUP SPARE with the PHYS option, enter into the defect log those defective sectors listed by LISTDEFECTS but not yet shown by LISTLOG. You should save a copy of your most current LISTLOG output to use when you next perform a PUP FORMAT, in case the lifetime defect log becomes unreadable.

- Table 5-3 describes the function of the FORMAT command according to the options used with it.

Table 5-3. PUP FORMAT Command Options and Their Functions

PUP FORMAT Command	Flags Those Sectors
FORMAT	that fail this FORMAT already flagged as bad
FORMAT /IN <filename>/	that fail this FORMAT already flagged as bad in the IN <filename> in the defect log
FORMAT NORETAIN	that fail this FORMAT
FORMAT /IN <listfile>/ NORETAIN	that fail this FORMAT in the IN <filename>

FORMAT Progress Report Listing

The display of the FORMAT progress report listing is shown in Figure 5-10.

```
DATE : <mmm dd yy> <hh:mm>
FORMAT OF $<device> [-P|-M] WITH <nnn> CYLINDERS  SYSTEM \<name>

                  ELAPSED  CURRENT  PROJECTED
CYLINDER  % DONE   TIME     TIME      TIME

    .        .       .        .         .
    .        .       .        .         .
    .        .       .        .         .

FORMAT COMPLETED, DATE : <mmm dd yy> <hh:mm>
```

Figure 5-10. FORMAT Progress Report

The FORMAT progress report headers are identified as follows:

DATE	is the date when the FORMAT progress report was initiated. (<mmm dd yy> = month, day, year; <hh:mm> = hour and minute)
$<device> [-P|-M]	is the name of the device (primary or mirror) on which the FORMAT progress report was performed.
<nnn> CYLINDERS	is the number of cylinders on that device.
SYSTEM \<name>	is the name of the system on which the device is being formatted.
CYLINDER	is the number of the cylinder. (Each cylinder has a unique number.)
%DONE	indicates the percentage of the disc that has been formatted to this point.
ELAPSED TIME	indicates how much time has elapsed since the progress report started.
CURRENT TIME	indicates the time at which this report line was generated.
PROJECTED TIME	indicates how much time it will take to complete the progress report.
FORMAT COMPLETED, DATE	shows the date on which the FORMAT operation on this particular volume was completed (month, day, year, hour, and minute).

FORMAT Execution Times

Table 5-4 gives approximate formatting times for various types of discs. All time estimates are based on a system where only an operating system and a PUP process are running.

Table 5-4. Disc Format Execution Times

Disc Drive	Subtype	Approximate Formatting Time
160MB (4103)	2	20 minutes
240MB (4104)	3	30 minutes
64MB (4105/4106)	4	17 minutes
64MB (4109) Winchester	5	10 minutes
540MB (4116) fixed media Winchester)	6	90 minutes
1.45MB (4109) fixed head Winchester	7	5 minutes
128MB (4110/4111)	8	20 minutes
264MB 4114/4115	9	35 minutes

Examples

1. To format the disc named $DATA, enter:

```
:PUP FORMAT $DATA
```

   FORMAT flags those sectors on the disc that were previously flagged as defective as well as those sectors that fail the FORMAT. The defect log is not used.

2. To format the primary volume of the mirrored pair named $SYSTEM and add at terminal $TERM1 a list of sector addresses to be declared defective, enter:

```
:PUP FORMAT /IN $TERM1/ $SYSTEM-P
```

   FORMAT flags those sectors on the disc that were previously flagged as defective as well as those sectors that fail the FORMAT. In addition, FORMAT flags those sectors listed in the defect log as well as any sectors typed in at terminal $TERM1.

3. To format with the NORETAIN option an unlabeled, demountable volume (a volume named "?" when listed by the LISTDEV command) whose logical device number is $11, enter:

```
:PUP FORMAT $11-P, NORETAIN
```

FORMAT with the NORETAIN option formats and tests all sectors on the disc, and flags those that fail the FORMAT. Any sectors marked as defective at an earlier time that pass this FORMAT are returned to service, regardless of whether they were spared and are in the lifetime defect log. FORMAT NORETAIN does not use the defect log.

4. To reformat the disc named $DATA, without regard for the current state of defect log, and to flag as defective those sectors listed in the file LISTBAD, as well as those found during the format, enter:

```
:PUP FORMAT / IN LISTBAD, OUT / $DATA, NORETAIN
```

HELP Command

The HELP command displays all PUP command names or the syntax of a particular PUP command.

The syntax of the HELP command is:

```
HELP [ / OUT <listfile> / ] [ <command-name> | ALL ]

<listfile>

   is the output device for the listing.  If you omit
   <listfile>, the listing prints on your home terminal.

<command-name>

   is the name of a PUP command about which HELP returns
   syntax information.

ALL

   causes PUP to return a list of all PUP commands.
```

Examples

1. To display on your terminal a list of all the PUP command names, this command produces the following display:

```
:PUP HELP ALL

ADDRTOCYL       ADDRTOFILE      ALLOWOPENS      CHECKSUM        COMMENT
CONSOLE         COPY            CYLTOADDR       DEMOUNT         DOWN
EXIT            FIXMICROCODE    FORMAT          HELP            LABEL
LISTBAD         LISTCACHE       LISTDEFECTS     LISTDEV         LISTFREE
LISTHEADERS     LISTLOG         LISTSPARES      LOADMICROCODE   PRIMARY
REBUILDDFS      REFRESH         REMOVE          REMOVEAUDITED   RENAME
REPLACEBOOT     REVIVE          SETCACHE        SPARE           STATUS
STOPOPENS       UP              UPDATEREV
```

2. To display on your terminal the syntax information about the PUP DOWN command, enter:

```
:PUP HELP DOWN
```

The following display appears:

```
DOWN [ ! ] <device> [, HARD ]

    ! is:

      Places the device down even if files are open
      on the device.

    <device> is:

        $<volume name>[-P|-M|-B|-MB]
        $<device name>
        $<logical device number>[-P|-M|-B|-MB]
```

3. To print, on a printer named $LP, syntax information about the PUP SPARE command, enter:

```
:PUP HELP /OUT $LP/ SPARE
```

LABEL Command

The LABEL command writes a volume label on a newly formatted disc pack or labels a previously labeled disc pack.

The LABEL command also allows you to set the cache configuration for a DP2 volume at the time of volume creation.

Optionally, the LABEL command then transfers operating system images and files from the system image tape (SIT tape) onto the disc volume.

CAUTION

Execution of the LABEL command destroys any existing files on the volume being labeled. Do not use this command to consolidate disc free space on $SYSTEM discs.

The syntax of the LABEL command is:

```
LABEL [ ! ] $<volume-name> , $<ldev-number>
        {                                                    }
        { , <max-num-of-files>                               }
        { , $<tape-file>                                     }
        { , <files-per-directory-extent>                     }
        {             [ , <block-size> = <num-blocks> ... ]  }
```

!

means PUP is to proceed with the LABEL operation even if files are open on the volume to be labeled.

$<volume-name>

is the name to be assigned to the new volume. If the disc pack is not demountable, the $<volume-name> must match the one assigned at system generation time.

→

$<ldev-number>

is the logical device number of the disc unit on which the volume is mounted.

<max-num-of-files>

is an estimate of the number of files allowed to exist on the volume. Specify this parameter for a DP1 volume only. The range is 1 to 524,280 (the size of the largest possible extent). Additionally, the number of files specified cannot produce a directory extent that exceeds the disc size.

$<tape-file>

is either the name or the logical device number of a magnetic tape unit containing the system image tape (SIT tape) to be copied. Specify this parameter to label the volume and then copy the contents of the tape to the volume.

<files-per-directory-extent>

is the directory extent-size specification for a DP2 disc only. The range is 1 to 262,140 (the size of the largest possible extent). Additionally, the number of files specified cannot produce a directory extent that exceeds the disc size.

Since the DP2 file directory is dynamically extendible (that is, it has multiple extents), you can use this parameter to define how many files should fit into each directory extent.

<block-size> = <num-blocks>

specifies the cache configuration for a DP2 volume as a string of "keyword = value" pairs where the "keyword" (that is, <block-size>) is a particular cache size, and "value" (<num-blocks>) is the value of the desired allocation. See "Considerations" and "Examples."

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- Execution of the LABEL command causes the volume to be in an up state at the conclusion of the operation.
- The name assigned by the LABEL command remains with the pack until it is explicitly relabeled by another LABEL command or renamed by the RENAME command.
- If an A03 or later version of the PUP LABEL command has been performed on a disc, then an earlier version (prior to A03) of the PUP REBUILDDFS command cannot be used on that disc.
- The LISTFREE command shows you the current directory size. Use that value in estimating the <max-num-of-files> or <files-per-directory-extent> value in the LABEL command.
- For a DP2 disc, the number of files per directory can be larger than for DP1 since the directory file is dynamically extendible.
- A valid DP2 <block-size> for a controller is a power-of-two multiple of the sector size, up to the maximum transfer size. You may specify <block-size> in bytes or kilobytes:

512 bytes (sector size)	or	.5K
1024	or	1K
2048	or	2K
4096 (maximum transfer size)	or	4K

- The maximum number of cache buffers that can be allocated for each DP2 cache is 8192. The minimum number is one.
- If your DP2 cache configuration does not allow at least two blocks for cache for each disc process in the volume disc-process group, the disc process overrides your requested values with the minimum number of blocks required. This is done to ensure that the cache is adequate to satisfy any request requiring a given buffer size.
- To avoid wasting unused page fragments, DP2 rounds up all allocations to the next whole page (for example, 512 = 6 becomes 512 = 8, or 2 pages). While the pages underlying the cache are not swappable, they are available to the memory manager when not in use.

- When a DP2 LABEL completes, a message of the following type appears:

  THE SYSTEM TABLES CONSUME n% OF THE AVAILABLE DISC SPACE.
  n% (n PAGES) OF THE DISC SPACE IS AVAILABLE FOR USER FILES.

- If user-requested allocations exceed the physical memory that can be used by the disc process, DP2 trims back the allocations until they fit. The amount of physical memory that can be used by the disc process depends on the installation, but in no case can it be more than 10 megabytes.

Examples

1. To write a volume label on the formatted disc pack named $NEWVOL, mounted on logical device number $7, and then copy the system image from the tape mounted on $TAPE onto $NEWVOL, enter:

   ```
   :PUP LABEL $NEWVOL, $7, $TAPE
   ```

2. To write a volume label on the disc pack named $DATA, mounted on logical device number $3, to receive a copy of the SIT tape currently on $TAPE4, enter:

   ```
   :PUP LABEL $DATA, $3, $TAPE4
   ```

3. If the disc $DATA in example 2 is a DP1 disc, you can designate 2000 as the maximum number of files to exist on the disc by entering:

   ```
   :PUP LABEL $DATA, $3, 2000
   ```

4. To write a volume label on the DP2 disc $DATA, mounted on logical device number 3, that holds 200 files for each extent, with 20 cache buffers for the smallest sector size and 40 cache buffers for the other sector sizes, enter:

   ```
   :PUP LABEL $DATA, $3, 200, 512=20, 1024=40, 2048=40, 4096=40
   ```

5. You can also designate the cache block size from example 3 in kilobytes:

   ```
   :PUP LABEL $DATA, $3, 200, .5K=20, 1K=40, 2K=40, 4K=40
   ```

LISTBAD Command

The LISTBAD command lists the disc sectors detected by the operating system as bad (that is, defective) but not yet spared, and the associated files on the specified disc volume.

The syntax of the LISTBAD command is:

```
LISTBAD [ / OUT <listfile> / ] { $<volume-name>  [-P]
                               {                 [-M]
                               { $<ldev-number>  [-P]
                               {                 [-M]
```

<listfile>

specifies the destination device for the listing. If omitted, the LISTBAD command directs the output to the home terminal.

$<volume-name>

is the name of the volume whose bad sectors are to be listed.

$<ldev-number>

is the logical device number of the volume whose bad sectors are to be listed.

-P | -M

specifies that the listing is for the primary or mirror device. -P or -M must be specified if the disc is a mirrored volume.

Considerations

- A sector is detected as bad by an I/O driver when an uncorrectable data error occurs. The operating system puts the defective sector on the spare tracks table.
- The LISTBAD command lists those disc areas that the operating system believes to be bad, but that the operator has not yet spared. Once spared (using PUP SPARE), these areas are removed from the spare tracks table and no longer appear in the LISTBAD report.

Listing Format

The format of the LISTBAD display listing is shown in Figure 5-11.

```
 CYL   HEAD  SECTOR     BYTE ADDR   TOSVERSION  TERMCODE
                        MSW    LSW
  .     .      .         .      .        .          .
  .     .      .         .      .        .          .

 FILE NAME            FILE ADDRESS          CYL   HEAD  SECTOR

    .                    .                   .     .     .
    .                    .                   .     .     .
```

Figure 5-11. LISTBAD Listing Format

Listing headers are identified as follows:

CYL HEAD	specify a track on $<volume-name>.
SECTOR	specifies a sector on $<volume-name>.
BYTE ADDR	specifies a logical byte address of a sector marked bad on $<volume-name>. MSW (most significant word) and LSW (least significant word) are given in octal notation.

TOSVERSION	specifies the version of the GUARDIAN operating system that detected the error on the disc (usually the GUARDIAN operating system) and its version number.
TERMCODE	indicates the device status, including the termination code in bits 1 through 7, that the disc controller returned to the source program.
FILE NAME	specifies the file associated with a sector that has been marked as being bad.
FILE ADDRESS	specifies the relative byte address in the file name associated with a sector that has been marked as bad.
CYL HEAD SECTOR	repeat the cylinder, head, and sector numbers given at the beginning.

The LISTBAD command output indicates the operating system version numbers under which the errors were detected and the termination codes associated with those errors.

Examples

1. To print on the printer named $LLP the list of bad sectors and their associated files on the volume named $DATA, enter:

```
:PUP LISTBAD /OUT $LLP/ $DATA
```

2. To print on your terminal the list of bad sectors and their associated files on the primary volume of the mirrored pair named $RIFLE, enter:

```
:PUP LISTBAD $RIFLE-P
```

The following information is displayed:

```
  CYL    HEAD    SECTOR      BYTE ADDR        TOSVERSION    TERMCODE
                             MSW    LSW

   662    14        28    %014030.060000    GUARDIAN A03   %004400

  FILE NAME        FILE ADDRESS          CYL   HEAD   SECTOR

  BIG.BIG        %014030.052340          662    14      28
```

LISTCACHE Command

LISTCACHE allows you to examine the current cache configuration for a volume. This command is available for DP2 discs only.

The syntax of the LISTCACHE command is:

```
LISTCACHE [ / OUT <listfile> / ]  { $<volume-name> }
                                  { $<ldev-number> }

          , { STAT[ISTICS] }
            { INIT[IALIZE] }
```

<listfile>

specifies the destination device for the listing. If omitted, LISTCACHE directs the output to the home terminal.

$<volume-name>

is the name of the volume whose cache configuration is to be examined.

$<ldev-number>

is the logical device number of the volume whose cache configuration is to be examined.

STAT[ISTICS]

prints out cache performance statistics and counters.

INIT[IALIZE]

allows you to reset the counters used to compute cache performance statistics printed by the STATISTICS option.

You must be a member of the super group to specify the INIT option.

Considerations

- If you attempt to use LISTCACHE on a DP1 disc, PUP sends you this message:

  THE 'LISTCACHE' COMMAND ONLY WORKS ON A DP2 DISC

- Disc cache configuration is the number and size of sectors read from disc and stored in the processor for use as virtual memory.

- If the number of cache read faults is large, use SETCACHE to reduce the number of cache blocks.

- In general, the cache should be adjusted so that cache read hits are high and cache read misses are low, but this decision is application dependent.

- Because an audit force involves several steps, it is time consuming; a high value in a LISTCACHE display indicates insufficient cache memory. Increase the blocks allocated with a SETCACHE command, assuming you have enough memory to support the increase.

- Ideally the number of writes for each control point should be zero. Refer to the Transaction Monitoring Facility (TMF) System Management and Operations Guide for further information on control points.

Listing Format

LISTCACHE displays both the user-requested and the system-assigned values, along with the total amount of memory allocated for the disc caches.

The format of the LISTCACHE display is shown in Figure 5-12.

```
DATE : mmm dd yyyy,  hh:mm
CACHE CONFIGURATION : $<volume-name>     SYSTEM : \<system-name>

CACHE BLOCK SIZE:    512          1024         2048          4096

BLOCKS REQUESTED:     n            n            n             n
BLOCKS ALLOCATED:     n            n            n             n

BYTES ALLOCATED TO CACHE:    nnnn K
```

Figure 5-12. LISTCACHE Listing Format

Listing elements are identified as follows:

DATE	is the date and time the LISTCACHE display was produced, in the form day-month-year, hour-minute.
CACHE CONFIGURATION	is the system and volume of this LISTCACHE.
CACHE BLOCK SIZE	lists the column headers for the block sizes.
BLOCKS REQUESTED	lists the number of blocks that were requested in the SETCACHE command or that were set by default.
BLOCKS ALLOCATED	lists the number of blocks allocated by SETCACHE.
BYTES ALLOCATED TO CACHE	is the total memory allocated to cache, equal to the sum of (blocks allocated times each cache block size). This number cannot exceed 6000 kilobytes.

The format for the LISTCACHE display, with the STATISTICS option selected, is shown in Figure 5-13.

```
DATE :  mmm dd yyyy,   hh:mm
CACHE STATISTICS : $<volume-name>       SYSTEM : \<system-name>
COUNTERS INITIALIZED :  mmm dd yyyy, hh:mm
ELAPSED TIME :              nn DAYS  hh:mm
CACHE BLOCK SIZE :   512       1024       2048       4096

BLOCKS REQUESTED :     .          .          .          .
BLOCKS ALLOCATED :     .          .          .          .
BLOCKS DIRTY :        .%         .%         .%         .%

CACHE READ HITS :     .%         .%         .%         .%
CACHE READ FAULTS :   .%         .%         .%         .%
CACHE READ MISSES :   .%         .%         .%         .%

CACHE WRITES :        .%         .%         .%         .%
CACHE WRITE HITS :    .%         .%         .%         .%

CACHE CALLS :          .          .          .          .
AUDIT FORCES :         .          .          .          .

BYTES ALLOCATED TO CACHE:    nnnn K      WRITES/CONTROL POINT :      n.nn
```

Figure 5-13. LISTCACHE Listing Format with STATISTICS Option

Additional listing headers that appear when using the STATISTICS option are identified as follows:

CACHE STATISTICS	is the same as CACHE CONFIGURATION above.
COUNTERS INITIALIZED	shows the date and time when the disc was brought up or when the most recent LISTCACHE INIT was executed, in the form day-month-year, hour-minute.
ELAPSED TIME	shows the total time elapsed since the most recent LISTCACHE INIT was executed, in days, hours, and minutes.
BLOCKS DIRTY	gives the percentage of blocks allocated that are currently dirty.
CACHE READ HITS CACHE READ FAULTS CACHE READ MISSES	when summed, should equal 100 percent for each cache block size. Each percentage is relative to the total number of reads that occurred.

CACHE READ HITS	is the percentage of time that the requested block was found in cache memory. A high percentage indicates high activity by a sequential-access file of the indicated cache block size. A random-access file would not have a high a hit rate.
CACHE READ FAULTS	is the percentage of time there was no memory underneath the cache block (because the memory manager had taken it). The disc process treats this as a miss and brings the block in from disc.
CACHE READ MISSES	is the percentage of time the requested block was not found in cache and the disc process had to bring it in from disc. A 40% miss rate (to a 60% hit rate) is acceptable for randomly accessed key-sequenced files, but is high for sequentially accessed files.
CACHE WRITES	is the percentage of cache calls that were writes. The percentage of CACHE READS is 100 percent minus CACHE WRITES.
CACHE WRITE HITS	is the percentage of time the block needed for a write was found in cache and was already dirty, thus saving a write to disc.
CACHE CALLS	is the total number of CACHE READS and CACHE WRITES during the measurement interval.
AUDIT FORCES	is the number of times an audited dirty block required an audit-trail write to disc to make room for a new block to be read from disc.
WRITES/CONTROL POINT	is the number of buffers that had to be written to disc at the most recent control point. Refer to the Transaction Monitoring Facility (TMF) System Management and Operations Guide for further information on control points.

Examples

1. To display the current cache configuration on the volume named $DP2, enter:

```
:PUP LISTCACHE $DP2
```

   The display shows the number of blocks currently allocated to each cache block size:

```
DATE : DEC 19 1984,  08:33
CACHE CONFIGURATION : $DP2              SYSTEM : \EAST
CACHE BLOCK SIZE :   512       1024       2048       4096

BLOCKS REQUESTED :     2          2         20          2
BLOCKS ALLOCATED :     8          6         20          6
BYTES ALLOCATED TO CACHE:      74 K
```

2. When you add the STATISTICS option to example 1, the command and display become:

```
:PUP LISTCACHE $DP2, STATISTICS
```

```
DATE : DEC 19 1984,  08:34
CACHE STATISTICS : $DP2                 SYSTEM : \EAST
COUNTERS INITIALIZED : DEC 18 1984,  20:29
ELAPSED TIME :                0 DAYS,  12:05
CACHE BLOCK SIZE :   512       1024       2048       4096

BLOCKS REQUESTED :     2          2         20          2
BLOCKS ALLOCATED :     8          6         20          6
BLOCKS DIRTY :        0%         0%        50%         0%

CACHE READ HITS :      0%         0%        68%        84%
CACHE READ FAULTS :    0%         0%         0%         0%
CACHE READ MISSES :  100%       100%        32%        16%

CACHE WRITES :       100%       100%        17%        15%
CACHE WRITE HITS :     0%         0%         3%         0%

CACHE CALLS :         78         54     207065      74603
AUDIT FORCES :         0          0          0          0

BYTES ALLOCATED TO CACHE:      74 K      WRITES/CONTROL POINT :      0.00
```

Note that the files accessed most frequently are of 2K cache block size and are probably random-access files. In contrast, the 4K cache block size files, which also had some activity, are probably sequentially accessed files, as reflected by their higher hit rate.

LISTDEFECTS Command

LISTDEFECTS scans an entire disc, a specific cylinder, or a particular head on a specified cylinder for sectors whose headers have been flagged as defective by a FORMAT or SPARE operation.

The syntax of the LISTDEFECTS command is:

```
LISTDEFECTS [ / OUT <listfile> / ] { $<volume-name>  [-P]
                                   {                 [-M]
                                   {
                                   { $<ldev-number>  [-P]
                                   {                 [-M]

            [ , cylinder [ , head ] ]
```

<listfile>

specifies the destination device for the listing. If the destination device is a terminal, LISTDEFECTS displays a full screen of information at a time, then asks whether the display should be continued. If omitted, LISTDEFECTS directs output to the home terminal.

$<volume-name>

is the name of the volume whose defective sectors are to be listed.

$<ldev-number>

is the logical device number of the volume whose defective sectors are to be listed.

-P | -M

indicates that the defective sectors are to be listed for the primary or mirror disc. -P or -M must be specified if the disc is a mirrored volume.

→

<cylinder>

is a cylinder number.

<head>

is a head number.

Listing Format

The format of the LISTDEFECTS display listing is shown in Figure 5-14.

```
PCYL   PHEAD   PSECT   SYNC%   FLAG%   CYL   HEAD   SECT
 .      .       .       .       .      .     .      .
 .      .       .       .       .      .     .      .
```

Figure 5-14. LISTDEFECTS Listing Format

Listing headers are identified as follows:

PCYL PHEAD PSECT	The physical cylinder, physical head, and physical sector make up a physical sector address on the disc.
SYNC% FLAGS%	These two items provide additional information about the faulty sector to Tandem Customer Engineers.
CYL HEAD SECT	The logical cylinder, head, and sector make up a logical sector address on the disc. If sector slipping has occurred because of a defect, the logical and physical addresses can differ. (Refer to the discussion on "Sector Sparing" in the System Operator's Guide for more information about sector slipping.)

Example

To display on your terminal the defective sectors on the primary volume of the mirrored pair named $DATA, enter:

```
:PUP LISTDEFECTS $DATA-P
```

LISTDEV Command

The LISTDEV command lists configuration characteristics of devices on the system.

The syntax of the LISTDEV command is:

```
LISTDEV [ / OUT <listfile> / ]

        [ $<device-name>                                          ]
        [ $<volume-name>                                          ]
        [ $<ldev-number>                                          ]
        [ <cpu> [ , <controller> ] [ , <type-spec> ]              ]
        [ <type-spec>                                             ]

        [ DETAIL [ STATUS <status-interval> ] ]
```

<listfile>

is the destination device for the listing. If omitted, the listing is sent to your home terminal.

NOTE

If none of the next five parameters is given (that is, if no devices are specified), the characteristics of all devices on the system are listed.

$<device-name>

is the name of the device whose configuration characteristics are to be listed.

$<volume-name>

is the name of the volume whose configuration characteristics are to be listed.

→

$<ldev-number>

is the logical device number of the device whose configuration characteristics are to be listed.

<cpu>

is a processor number; the characteristics of the devices connected to the specified processor are to be listed.

<controller>

is a controller number; the characteristics of the devices connected to the specified processor (see <cpu>) and controller are to be listed.

<type-spec>

specifies a particular device type for which characteristics are to be listed. <type-spec> is one of the following:

```
DISC    [ <subtype> ]
TERM    [ <subtype> ]
TAPE    [ <subtype> ]
CARD    [ <subtype> ]
PRINTER [ <subtype> ]
TYPE <type-num> [ , <subtype> ]
```

DISC [<subtype>]

lists the characteristics of all discs or, optionally, all discs of a specific <subtype>.

TERM [<subtype>]

lists the characteristics of all terminals or, optionally, all terminals of a specific <subtype>.

→

TAPE [<subtype>]

lists the characteristics of all magnetic tapes or, optionally, all tapes of a specific <subtype>.

CARD [<subtype>]

lists the characteristics of all punched card readers or, optionally, all card readers of a specific <subtype>.

PRINTER [<subtype>]

lists the characteristics of all line printers or, optionally, all printers of a specific <subtype>.

TYPE <type-num> [, <subtype>]

lists the characteristics of all devices of a specific <type-num> and, optionally, <subtype>.

<type-num>

is an integer type number.

<subtype>

is an integer subtype number.

DETAIL

for any disc with a revive in progress (see the REVIVE command later in this section), shows additional information about what cylinder, how many tracks, and the time interval and projected completion time of the revive.

→

STATUS [<status-interval>]

causes the LISTDEV display to be repeated at periodic intervals. The integer range for <status-interval> is 1 through 60, specifying the number of minutes between progress reports. If <status-interval> is not specified, the default is five minutes. To terminate this listing, press the BREAK key.

Consideration

If a device does not respond within 60 seconds to a LISTDEV request for information, the following message appears on your terminal:

WARNING, DEVICE INFORMATION REQUEST TIMED OUT (ERROR 40) ON $<ldev-num> (PID <cpu>,<pin>)

If this timeout occurs on a primary device, PUP declares it the backup device and displays it as such in the LISTDEV display, with an "I" (for "inaccessible") in the STATE field (see Figure 5-15). Conversely, the real backup device appears on the LISTDEV display as the primary.

If the cause of the timeout is removed or goes away (extreme congestion, for example), then another LISTDEV succeeds without any error message, and the primary and backup devices appear in their normal positions.

Listing Format

The format of the LISTDEV display listing without the DETAIL option is shown in Figure 5-15.

LDEV	NAME	STATE	PPID	PCU	BPID	BCU	DMNT	TYPE	SUBTYPE	RECSIZE	DPTYPE
.	.	.	.	.	.	.	.	.	.	.	(for
.	.	.	.	.	.	.	.	.	.	.	discs
.	.	.	.	.	.	.	.	.	.	.	only)

Figure 5-15. LISTDEV Listing Format

Listing headers are identified as follows:

LDEV is a logical device number.

NAME is either a device name or a question mark if a phantom disc (that is, a demountable volume is not mounted).

The following forms are shown only for mirrored disc volumes and some data-communications devices:

$<volume-name>-P is displayed if the device is the primary device of a mirrored disc, or if it is the write device of a synchronous controller.

$<volume-name>-M is displayed if the device is the mirror device.

$<volume-name>-B is displayed if the device is the backup controller path to the primary device of a mirrored disc or is the read device of a synchronous controller.

$<volume-name>-MB is displayed if the device is the backup controller path to the mirror device.

STATE indicates the current state of the device in the primary process of the device, where:

" " = up
"D" = down
"H" = hard down state; that is, the device is down due to a hardware error or has been placed in this state by the PUP DOWN command.
"I" = inaccessible (primary or backup CPUs are down)

"R" = revive in progress
"S" = special request state, meaning that the disc is being used by PUP or some other privileged process.
"*" = preferred path to a dual-port disc
"ON" = operator disc logging on (displayed for $0 only--the operator process, LDEV 1)
"OFF" = operator disc logging off (displayed for $0 only--the operator process, LDEV 1)

PPID is the <cpu>,<pin> (CPU number and process ID number) of the current primary process of a device.

PCU is the controller number and unit number under control of the current primary process. For the operator process, this is the logical device number of the operator console.

BPID is the <cpu>,<pin> (CPU number and process ID number) of the current backup process of a device.

BCU is the controller number and unit number under control of the current backup process.

DMNT indicates whether or not a volume is demountable.

TYPE is the type number of the device, as given in Table 5-5. If an "A" appears next to the TYPE number, the volume is audited by the Transaction Monitoring Facility (TMF).

SUBTYPE is the subtype number of the device, as given in Table 5-5.

RECSIZE is either the configured physical record size of a nondisc device or the maximum transfer length of a disc.

DPTYPE "DP2" in this column specifies a volume configured for DP2, and "DP1" specifies a volume configured for DP1. If the disc is demountable, "DP1" or "DP2" is printed dependent upon how the disc is configured at system generation. This column prints when you specify LISTDEV disc information, that is:

```
LISTDEV DISC,
LISTDEV $<volume-name>,
LISTDEV $<volume-number>, or
LISTDEV TYPE 3.
```

Table 5-5. Device Types and Subtypes--LISTDEV Output
(Continued on next page)

TYPE	SUBTYPE
0 = Process	0
1 = Operator console ($0)	0
2 = $RECEIVE	0
3 = Disc	3 = 240 MB capacity (P/N 4104) 4 = 64 MB capacity (P/N 4105/4106) 5 = 64 MB capacity (P/N 4109) 6 = 540 MB capacity (P/N 4116) 7 = 1.45 MB fixed head portion (P/N 4109) 8 = 128 MB (P/N 4110/4111) 9 = 264 MB capacity (P/N 4114/4115)
4 = Magnetic tape	0 = Nine-track 1 = Seven-track 2 = Tri-density tape drive (P/N 5106)
5 = Line printer	0 = Belt printer 1 = Drum or band 2 = Current-loop, belt type 3 = Matrix serial (P/N 5508) 4 = Matrix serial (P/N 5520) 5 = Band, extended character set 6 = Letter-quality printer (P/N 5530) 32 = DTR printer
6 = Terminal	0 = Conversational mode 1 = Page mode (P/N 6511/6512) 2 = Page mode (P/N 6520/6524) 3 = Page mode (Remote 6520) 4 = Page mode (P/N 6530) 5 = Page mode (Remote 6530) 6-10 = Conversational mode (various screen sizes) 32 = Hard-copy console
(SNAX ITI protocol)	20 = 3277/3278 model 1 (screen size 12x40) 21 = 3277/3278 model 2 (screen size 24x80) 22 = 3278 model 3 (screen size 32x80) 23 = 3278 model 4 (screen size 43x80) 24 = 3278 model 5 (screen size 27x132)
7 = ENVOY data communication line	0 = Bisync, point-to-point, nonswitched 1 = Bisync, point-to-point, switched 2 = Bisync, multipoint, tributary 3 = Bisync, multipoint, supervisor 8 = ADM-2, multipoint, supervisor 9 = TINET, multipoint, supervisor 10 = Burroughs, multipoint, supervisor 13 = Burroughs, point-to-point contention 30 = Full duplex, out line 31 = Full duplex, in line 40 = Asynchronous line supervisor 50 = Isochronous line 56 = Auto-call unit
8 = Card reader	0
9 = Process-to-process interface	0 = X25AM Process

Table 5-5. Device Types and Subtypes--LISTDEV Output (Continued)

TYPE		SUBTYPE	
10	= 3277 CRT mode interface	1-5	= Page mode, various screen sizes
		6	= 328x printers
	(SNAX CRT protocol)	20	= 3277/3278 model 1 (screen size 12x40)
		21	= 3277/3278 model 2 (screen size 24x80)
		22	= 3278 model 3 (screen size 32x80)
		23	= 3278 model 4 (screen size 43x80)
		24	= 3278 model 5 (screen size 27x132)
		30	= 328x printer (using CRT protocol)
11	= ENVOYACP Data Communication Line	40	= Synchronous data link control (SDLC)
		41	= High-level data link control (HDLC)
		42	= Advanced data communication control protocol (ADCCP)
12	= TIL (Tandem to IBM Link)	0	
13	= SNAX	5	= Service manager
20-23	= TMF (Transaction Monitoring Facility)	0	
26	= THL (Tandem HyperLink)	0	
27	= FOX Interprocessor BUS Monitor ($IPB)	0	
50	= CSM (Communication Subsystem Monitor for 6100 subsystem)	0	
51	= CP6100	1	= BISYNC
		2	= ADCCP
		3	= TINET
52	= INFOSAT	0	= Earth station control line
		2	= Satellite connect
53	= CSM (ATP 6100)	0	= Terminals attached to 6100 subsystem (CLIP1)
		1	= CLIP4
58	= SNAX	0	= SDLC Line
59	= Data communication line (AM6520)	0	= Line to controller
		10	= Line to 6100 subsystem
60	= Data communication line	0	= AM3270 line to controller
		1	= TR3271 line to controller
		10	= Line to 6100 subsystem
61	= X.25 data communication line	0-62	= Line to controller (any subtype is accepted)
		63	= Line to 6100 subsystem
62	= EXPAND Network Control Process (NCP)	0	

Table 5-5. Device Types and Subtype - LISTDEV Output (Concluded)

TYPE	SUBTYPE
63 = EXPAND line handler	0 = Single-line handler for direct-connect synchronous line and satellite-connect synchronous line 1 = Multiline path handler for path 2 = Multiline path handler for direct-connect synchronous line and satellite-connect synchronous line 3 = FOX line handler 5 = Single-line handler for direct-connect 6100 subsystem 6 = Multi-line handler for direct-connect 6100 subsystem

The format of the LISTDEV display listing with the DETAIL option is shown in Figure 5-16.

```
LDEV NAME STATE PPID  PCU   BPID  BCU  DMNT TYPE SUBTYPE  RECSIZE  DPTYPE

 .    .     .     .    .     .     .    .    .      .        .      (for
 .    .     .     .    .     .     .    .    .      .        .      discs
 .    .     .     .    .     .     .    .    .      .        .      only)

 REVIVE CURRENTLY AT CYLINDER <nnn> OF <nnn> (%DONE)
 TRACKS PER INTERVAL = <nn> , INTERVAL = <nnn> * 10 MILLISECONDS
```

Figure 5-16. LISTDEV Listing Format with DETAIL

The additional listing headers that appear when you specify the DETAIL option are identified as follows:

REVIVE CURRENTLY AT CYLINDER <nnn> OF <nnn> (%DONE)	indicates that the REVIVE operation is currently at the specified cylinder number <nnn>; out of <nnn> total cylinders. (%DONE) gives the percentage of the cylinder that has been revived.
TRACKS PER INTERVAL = <nn>	indicates how many tracks are revived for each interval.
INTERVAL = <nnn> * 10 MILLISECONDS	indicates the interval between each COPY operation and the next.

Examples

1. To display on your terminal a list of all configured type-3 devices, enter:

```
:PUP LISTDEV TYPE 3
```

The following information is returned:

```
LDEV  NAME      STATE  PPID    PCU   BPID    BCU   DMNT TYPE SUBTYPE RECSIZE  DPTYPE

 4    $SYSTEM-P   *   00,004  %10   01,004  %10          3 A   2       4096    DP2
      $SYSTEM-B       00,004  %110  01,004  %110
      $SYSTEM-M   *   00,004  %11   01,004  %11
      $SYSTEM-MB      00,004  %111  01,004  %111
 5    $DATA-P     *   00,005  %114  01,005  %114   Y     3       3       4096    DP2
      $DATA-B         00,005  %14   01,005  %14
 6    $SPOOL-P    *   02,006  %15   02,003  %15    Y     3       6       4096    DP1
      $SPOOL-B        02,006  %115  02,003  %115
```

Notice the "A" by the TYPE number of $SYSTEM. This indicates that $SYSTEM is a volume audited by TMF.

Note also that $SPOOL is a DP1 volume, while $SYSTEM and $DATA are DP2 volumes.

2. To display on your terminal a list of all configured devices of type 5, subtype 1, enter:

```
:PUP LISTDEV TYPE 5, 1
```

```
LDEV   NAME STATE  PPID    PCU    BPID    BCU   DMNT TYPE SUBTYPE RECSIZE

 37  $LPRMT1      02,009  %21    03,009  %21          5     1       132
 38  $LP          00,007  %21    01,007  %21          5     1       132
 39  $LLP         05,015  %121   04,015  %121         5     1       132
```

3. To print on the printer named $LP a list of all the devices connected to CPU 6, enter:

```
:PUP LISTDEV /OUT $LP/ 6
```

LISTFREE Command

The LISTFREE command lists the number of free pages, the sizes of contiguous blocks of free pages, the number of files, and the maximum number of files on a disc volume.

The syntax of the LISTFREE command is:

```
LISTFREE [ / OUT <listfile> / ] [ $<volume-name> ]
                               [ $<ldev-number> ]

         [ , HIST [ , <intervals> ] ]
```

<listfile>

specifies the output device for the listing. If you omit <listfile>, the LISTFREE command directs its output to the home terminal.

$<volume-name>

is the name of the volume whose free space is to be listed. If omitted, the free space on the default volume is listed.

$<ldev-number>

is the logical device number of the volume whose free space is to be listed. If omitted, the free space on the default volume is listed.

HIST

causes the free-space data to be listed in the form of a histogram showing the frequency of free-space fragments by size. If omitted, the free-space data is listed by blocks of pages in order of disc address.

→

<intervals>

is an ascending sequence of integers, separated by commas, that represent sizes of the blocks of data. These integers are used as the limits of the histogram intervals. The rules for specifying <intervals> are:

Integers must be in ascending sequence

a(1) ,a(2) ,... ,a(n)

where a and n are defined as:

n <= 10, indicating that 1 to 10 integers can be specified

a(i) < a(i + 1), indicating that the first integer must be less than the second integer, the second integer must be less than the third integer, and so forth

1 <= a(i) <= 32767, indicating that no integer can be less than 1 or greater than 32767

For example:

```
:PUP LISTFREE $BOOKS1, HIST, 2, 8, 37, 39, 51
```

If HIST is specified but <intervals> is omitted, the sequence 2, 4, 8, ... , 1024 is used.

Consideration

Use the LISTFREE command to tell whether you have enough directory space to successfully complete a RESTORE operation. The LISTFREE command shows you the current directory size. Use that value in the LABEL command. Refer to the System Operator's Guide for the "Tape Method of Consolidating Disc Free Space."

Listing Format

The format of the LISTFREE display listing when HIST is not specified is shown in Figure 5-17.

```
    STARTING PAGE    # OF PAGES

           <num>       <num-pages>
             .             .
             .             .
    USED FILE PAGES=<num-pages>
    FREE FILE PAGES=<num-pages>    FRAGMENTS=<num-fragments>
    FILES=<num-files>              DIRECTORY SIZE=<num-files>
```

Figure 5-17. LISTFREE Listing Format

Listing headers are identified as follows:

STARTING PAGE	is the logical byte address divided by 2048 of the first file page in a contiguous block of free file pages.
# OF PAGES	is the number of pages in the block of free space.

NOTE

PUP must read all file labels on the volume before the remaining part of the listing is displayed. This can take several minutes, depending on the number of files on the volume.

USED FILE PAGES	is the number of 2048-byte pages allocated to files.
FREE FILE PAGES	is the number of 2048-byte pages available for allocation to files.
FILES	is the number of files existing on the volume.

FRAGMENTS	is the number of free-space fragments on the volume.
DIRECTORY SIZE	is the estimated number of files that can exist on the volume at any one time. This may be slightly more than either the number of directory entries specified during a PUP LABEL or the number specified at system generation. This is because LABEL rounds up all space allocations to the next whole page. For more information on how the system interprets SYSTEM_VOLUME_DIRECTORY_SIZE in the SYSGEN Configuration File, see the System Management Manual.

Listing Format With HIST Option

The format of the LISTFREE display listing with HIST specified is shown in Figure 5-18.

```
DISK FREESPACE HISTOGRAM FOR VOLUME $<vol>, SYSTEM \<syst>

                                NUMBER              PERCENT
  FRAGMENT SIZE              OF FRAGMENTS     OF TOTAL FRAGMENTS

  BELOW     a(1)          <num-fragments>    <percent-fragments>
   a(1) thru a(2) -1              .                   .
          .                       .                   .
          .                       .                   .
 a(n-1) thru a(n) -1              .                   .
   a(n) AND UP                    .                   .
```

Figure 5-18. LISTFREE Listing Format with HIST Option

Listing headers are identified as follows:

FRAGMENT SIZE	is explained in the LISTFREE command syntax for <intervals>.
NUMBER OF FRAGMENTS	is the number of fragments of free space whose size (in pages) lies in the indicated range.
PERCENT OF TOTAL FRAGMENTS	is the percentage of the total number of fragments of free space represented by <num-fragments>.

Examples

1. To display on your terminal disc free-space information for the volume named $LOAN, enter:

```
:PUP LISTFREE $LOAN
```

The following information is returned:

```
 STARTING PAGE   # OF PAGES   VOLUME $BOOKS1 ,  SYSTEM \TSB

        6878              1
       10501              1
                .
                .
      113994              6

 USED FILE PAGES=107275
 FREE FILE PAGES=4854          FRAGMENTS=1200
 FILES=7764                    DIRECTORY SIZE=10000
```

2. To print on the device named $LLP disc free-space information for the logical device $7 with its histogram, enter:

```
:PUP LISTFREE /OUT $LLP/ $7, HIST, 1, 4, 7,160
```

LISTHEADERS Command

LISTHEADERS lists the sector headers for an entire disc, a specified cylinder, or a particular head on a specified cylinder. This is useful in determining the physical sector to which a logical address maps.

The syntax of the LISTHEADERS command is:

```
LISTHEADERS [ / OUT <listfile> / ] { $<volume-name> [ -P ]
                                   {               [ -M ]
                                   {
                                   { $<ldev-number> [ -P ]
                                   {                [ -M ]

          [ , <cylinder> [ , <head> ] ]
```

<listfile>

is the output device for the listing. If the output device is your home terminal, after displaying a full screen of information, LISTHEADERS asks whether you wish to continue the display.

$<volume-name>

is the name of the volume whose sector headers are to be listed.

$<ldev-number>

is the logical device number of the volume whose sector headers are to be listed.

-P | -M

specifies a listing for the primary or mirror device. -P or -M must be specified if the listing is for a mirrored disc.

→

<cylinder>

is the cylinder number of the specific cylinder for which sector headers are to be listed.

<head>

is the head number of the specific head on the <cylinder> for which sector headers are to be listed.

Consideration

The mapping displayed by this command reflects any "slipping" caused by use of the SPARE or FORMAT commands. Slipping occurs when a spare sector is available on the cylinder and a defective sector is being spared. The sectors immediately following the defective sector are renumbered; that is, each sector is assigned a new logical address. For example, if sector 1 is defective, sector 2 is assigned the logical address of 1, sector 3 is assigned 2, and so on.

Listing Format

The format of the LISTHEADERS display listing is shown in Figure 5-19.

```
PCYL   PHEAD   PSECT   SYNC%   FLAG%   CYL   HEAD   SECT
 .      .       .       .       .      .     .      .
 .      .       .       .       .      .     .      .
```

Figure 5-19. LISTHEADERS Listing Format

Listing headers are identified as follows:

PCYL PHEAD PSECT	The physical cylinder, the physical head, and the physical sector make up a physical sector address on the disc.
SYNC% FLAGS%	These two items provide additional information about the faulty sector to your Tandem representative.
CYL HEAD SECT	The logical cylinder, head, and sector numbers make up a logical sector address on the disc. If sector slipping has occurred because of a defect, the logical and physical addresses can differ. (Refer to the discussion on "Sector Sparing" in the System Operator's Guide for more information about sector slipping.)

Examples

1. To list on the device named $LP the sector headers for the mirror device of a disc named $DATA, enter:

```
:PUP LISTHEADERS /OUT $LP/ $DATA-M
```

2. To list and display on your terminal the sector headers for logical device $8, cylinder number 3, head number 0, enter:

```
:PUP LISTHEADERS $8,3,0
```

The following information is returned:

PCYL	PHEAD	PSECT	SYNC%	FLAG%	CYL	HEAD	SECT
3	0	0	360	350	3	0	0
3	0	1	360	350	3	0	1
3	0	2	360	350	3	0	2
			.				
			.				
3	0	20	360	354	3	0	20

LISTLOG Command

The LISTLOG command prints the lifetime defect log of defective disc sectors.

The syntax of the LISTLOG command is:

```
LISTLOG [ / OUT <listfile> / ] { $<volume-name> [-P] }
                               { $<ldev-number> [-M] }

        [ , <cylinder> [ , <head> ] ]
```

<listfile>

specifies the output device for the listing. If you omit <listfile>, LISTLOG directs its output to the home terminal.

$<volume-name>

is the name of the volume whose defect log is to be listed.

$<ldev-number>

is the logical device number of the volume whose defect log is to be listed.

-P | -M

specifies that the listing is for the primary or mirror device. -P or -M must be specified if the volume is a mirrored volume.

→

<cylinder>

limits the defect log listing to those defects listed for a specific cylinder.

<head>

limits the defect log listing to those defects listed for a specific head of the specified cylinder.

Considerations

- The defect log is created at Tandem Computers; it includes any defects that the disc vendor listed for Tandem or that Tandem detected during the initial formatting of the disc. The rest of the log consists of defective sectors that have been spared.
- The defect log is recorded in duplicate (that is, there are two copies) on the next-to-last cylinder. (The last cylinder is reserved for diagnostic use.) A hard-copy list of the initial defects is in an envelope attached to the disc drive. Fixed-head Winchester discs (disc subtype 7) do not have defect logs.
- The LISTDEFECTS command is different from the LISTLOG command. The LISTDEFECTS command scans the physical disc sectors for all defective sectors, whether spared or not; the LISTLOG command lists those defective sectors that have been entered into the defect log by the PUP SPARE command.

Listing Format

The format of the LISTLOG display listing is shown in Figure 5-20.

```
LIFETIME DEFECT LOG REPORT ON $<volume> <date> <time> <system>

 PCYL   PHEAD   PSECT      SOURCE            TERMINATION CODE

   .      .       .          .                    .
   .      .       .          .                    .
   .      .       .          .                    .
   .      .       .          .                    .

 THE DEFECT LOG CONTAINS <nn> DEFECTS
```

Figure 5-20. LISTLOG Listing Format

Listing headers are identified as follows:

PCYL PHEAD	make up the physical address of a track on $<volume>.
PSECT	is the defective sector within the track on $<volume>.
SOURCE	indicates the name of the program that detected the error on the disc, for example, the GUARDIAN operating system.
TERMINATION CODE	is the device status (including the termination code in bits 1 through 7) that the disc controller returned to the <source> program.
<nn>	indicates the total number of defects found in the defect log.

Example

To list and display the defect log for the mirror device of a mirrored volume named $SPOOL, enter:

```
:PUP LISTLOG $SPOOL-M
```

LISTSPARES Command

The LISTSPARES command lists, for a specified disc volume, the assigned extension cylinders and their associated files.

The syntax of the LISTSPARES command is:

```
LISTSPARES [ / OUT <listfile> / ] { $<volume-name> [-P] }
                                  { $<ldev-number> [-M] }
```

<listfile>

is the output device for the listing. If you omit <listfile>, LISTSPARES directs its output to the home terminal.

$<volume-name>

is the name of the volume whose spare tracks are to be listed.

$<ldev-number>

is the logical device number of the volume whose spare tracks are to be listed.

-P | -M

specifies that the listing is for the primary or mirror disc of a mirrored volume. -P or -M must be specified if the volume is a mirrored volume.

Considerations

- The LISTSPARES command indicates files that are undergoing performance degradation; it does not indicate the bad sectors on the disc (as the LISTDEFECTS and LISTBAD commands do). The LISTSPARES command lists extension cylinders. The cylinder, head, and sector displayed in this report is not one of the bad sectors on the cylinder; it is the first sector on the cylinder to experience performance degradation. The name of the file associated with this sector is also printed. (See "Disc Structure" in the System Operator's Guide for more information about cylinders.)

- If the LISTSPARES command finds no SPARE operations to carry out, it responds with the message:

  NO CYLINDERS HAVE BEEN EXTENDED

- An extension cylinder takes longer to access because the controller looks first at the header of the defective sector to find the location of the spare; then it must perform a seek to the extension cylinder. The LISTSPARES command is useful for determining which heavily used files are on extension cylinders. You can then use the File Utility Program (FUP) DUP command (described in Section 2) to relocate such files.

Listing Format

The format of the LISTSPARES display listing is shown in Figure 5-21.

```
    CYL       HEAD       SECTOR              BYTE ADDR
                                              MSW    LSW
<cylinder>  <head>     <sector>       <absolute-byte-address>
     .          .          .                    .
     .          .          .                    .

  FILE NAME          FILE ADDRESS              CYL   HEAD  SECTOR

 <filename>    <start-addr>-<end-addr>      <cyl> <head><sector>
     .                    .                    .     .     .
     .                    .                    .     .     .
```

Figure 5-21. LISTSPARES Listing Format

Listing headers are identified as follows:

CYL HEAD SECTOR	identify the first sector that lies in the extension.
FILE NAME	is the name of the file that partially lies in the extension.
FILE ADDRESS	gives the starting and ending relative byte addresses of the file region that lies in the extension.

Examples

1. To display on your terminal the first logical sector in an extension cylinder and its associated files on the primary device of a mirrored volume named $RIFLE, enter:

```
:PUP LISTSPARES $RIFLE-P
```

2. To print on the device named $LP the first logical sector in an extension cylinder and its associated files on the primary device of a mirrored volume named $SYSTEM, enter:

```
:PUP LISTSPARES /OUT $LP/ $SYSTEM-P
```

LOADMICROCODE Command

The LOADMICROCODE command requests the I/O process to find the appropriate microcode file and to download it to the specified downloadable controller.

The syntax of the LOADMICROCODE command is:

```
LOADMICROCODE [%] <cpu> , [%] <controller>
```

<cpu>

is a number that identifies one of the processors to which the downloadable controller is attached. If this parameter is preceded by a percent sign (%), PUP treats it as an octal number.

<controller>

is the number that identifies the controller. If this parameter is preceded by a percent sign (%), PUP treats it as an octal number.

General Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- All devices connected to the controller must be in the down state before executing the LOADMICROCODE command.

Considerations for the 3107 Downloadable Disc Controller

- The LOADMICROCODE command does not download microcode to the 3107 controller; it merely prepares the controller for downloading. The disc process automatically downloads the microcode the first time that a disc on the controller is accessed subsequent to the LOADMICROCODE command.
- The 3107 disc microcode to be downloaded is kept in resident memory.

Considerations for the 3206 Tri-Density Tape Controller

- The I/O process downloads the microcode command in response to the PUP LOADMICROCODE command.
- Associated with each downloadable controller is a disc subvolume named $SYSTEM.M<hpn> that contains all possible microcode files necessary for all hardware and PROM versions of the controller board to be loaded. The names of the individual files in this subvolume are:

  $SYSTEM.M<hpn>.<hhhppp>

  <hpn> is the highest assembly part number of the controller board to be loaded.

  <hhh> is the hardware revision number of the board.

  <ppp> is the PROM revision number.

- When you enter the LOADMICROCODE command, the operating system obtains from the controller the following information:

  --The highest assembly part number of the controller board

  --The associated hardware

  --PROM revision numbers

  Then it performs a search for a file named $SYSTEM.M<hpn>.<hhhppp> (the primary microcode file). This file contains the corresponding downloadable microcode that downloads the microcode to the controller.

- If the operating system cannot locate the primary microcode file, it attempts to open a backup microcode file named $SYSTEM.SYS<nn>.M<hpn>. (In the subvolume SYS<nn>, the value <nn> is the system subvolume number.) This second microcode file is intended for use with any version of the hardware. This backup file is placed on the system image tape created by the SYSGEN program and is copied to the system disc when that disc is loaded. (Refer to the System Management Manual for additional information on the system image tape and SYSGEN.)

  If an error occurs during a download operation, console message 100 (MICROCODE LOADING FAILURE) prints on the operator console. (See "Console Messages" in the System Messages Manual for details.) This message appears once if the operating system fails to access the primary microcode file and twice if it also fails to access the backup microcode file. If the failure occurred because the operating system was unable to locate either of the two microcode files, use the PUP STATUS command and the RESTORE program (described in Section 4) to restore the microcode file subvolume from your site update tape. Then issue the LOADMICROCODE command to download the microcode.

Example

To download the microcode file to controller number 5 attached to CPU number 3, enter:

```
:PUP LOADMICROCODE 3, 5
```

PRIMARY Command

The PRIMARY command has two functions:

- To make a designated processor module the primary path for a specified device
- To designate a preferred controller path for a dual-port disc

  This designates the controller to be used when operable (communication always occurs on the primary I/O channel in any case). This designation has no effect on the primary processor module designation.

The syntax of the PRIMARY command is:

```
PRIMARY [ ! ]  { $<device-name> , <cpu>                    }
               { $<ldev-number> , <cpu>                    }
               {                                           }
               { $<ldev-number> [ -P  ]                    }
               {                [ -M  ]                    }
               {                [ -B  ]                    }
               {                [ -MB ]                    }
               {                                           }
               { $<volume-name> [ -P  ]                    }
               {                [ -M  ]                    }
               {                [ -B  ]                    }
               {                [ -MB ]                    }
```

!

causes the disc process to give away ownership of all four controller paths. The default is to give ownership to only its -P and -M controllers.

$<device-name>,<cpu>

is the name of the device whose primary path is to be changed. The primary path is changed to the processor module specified by <cpu>. This value is an integer; it must identify one of the two processor modules to which the device is connected.

→

$<ldev-number>,<cpu>

is the logical device number of the device whose primary path is to be changed. The primary path is changed to the processor module specified by <cpu>. This value is an integer; it must identify one of the two processor modules to which the device is connected.

$<ldev-number>-P
$<volume-name>-P

designates that the primary controller is the preferred controller path to the primary device of the mirrored volume having the logical device number $<ldev-number> or the volume name $<volume-name>.

$<ldev-number>-M
$<volume-name>-M

designates that the primary controller is the preferred controller path to the mirror device of the mirrored volume having the logical device number $<ldev-number> or the volume name $<volume-name>.

$<ldev-number>-B
$<volume-name>-B

designates that the backup controller is the preferred controller path to the primary device of the mirrored volume having the logical device number $<ldev-number> or the volume name $<volume-name>.

$<ldev-number>-MB
$<volume-name>-MB

designates that the backup controller is the preferred controller path to the mirror device of the mirrored volume having the logical device number $<ldev-number> or the volume name $<volume-name>.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)

- When the primary processor path is switched for a device that is a member of a controller group, the primary processor paths for all devices in the group are switched.

- The switch to the backup controller for a disc is effective only if the current primary processor module has ownership of the backup controller. If this is not the case (as shown below for $DATA-P), the primary processor (00) does not have ownership of the backup controller.

LDEV	NAME	STATE	PPID	PCU	BPID	BCU	DMNT	TYPE	SUBTYPE	RECSIZE	DPTYPE
4	$SYSTEM-P	*	01,004	%10	00,004	%10		3	2	4096	DP2
	$SYSTEM-B		01,004	%110	00,004	%110					
	$SYSTEM-M	*	01,004	%11	00,004	%11					
	$SYSTEM-MB		01,004	%111	00,004	%111					
5	$DATA-P	*	00,005	%114	01,005	%114	Y	3	3	4096	DP1
	$DATA-B		00,005	%14	01,005	%14					

  The entire logical device (in this case LDEV 5) should be switched to the other processor, or the desired processor module should be given ownership of the backup controller. The latter can be accomplished by executing a PRIMARY command to another disc device that uses the controller as its primary controller (in this case LDEV 4), so it is running in the same processor as the disc above.

- Controller path selection does not survive a processor switch. When a processor switch occurs, the -P path (and the -M path if the disc is mirrored) is used. For example, a mirrored disc is running in processor 0 through the -B and -MB paths. The command PRIMARY $volume,1 is issued. The disc is now running in processor 1 through the -P and -M paths.

CAUTION

When configuring a 4109 disc, you must have SYSGEN entries for both the moving-head part and the fixed-head part. If these are not configured properly, the system hangs when you PUP PRIMARY a 4109 disc to its backup path. Refer to "4109 Controller Considerations" in the System Management Manual for more information.

Examples

1. To display on your terminal a list of the configured devices of type 3, enter:

```
:PUP LISTDEV TYPE 3
```

The following information is returned:

```
LDEV NAME   STATE  PPID    PCU   BPID    BCU  DMNT TYPE SUBTYPE RECSIZE  DPTYPE

4  $SYSTEM-P    *  00,004  %10   01,004  %10          3     2      4096    DP2
   $SYSTEM-B       00,004  %110  01,004  %110
   $SYSTEM-M    *  00,004  %11   01,004  %11
   $SYSTEM-MB      00,004  %111  01,004  %111
5  $DATA-P      *  00,005  %114  01,005  %114  Y      3     3      4096    DP1
   $DATA-B         00,005  %14   01,005  %14
```

In example 1, the asterisk that appears in the STATE column indicates the primary controller path for those specified volumes. To change the primary path for the nonmirrored volume $DATA and display the results, enter:

```
:PUP PRIMARY $DATA-B
:PUP LISTDEV $DATA
```

2. To change the primary controller path for the mirrored volume pair named $SYSTEM, enter:

```
:PUP PRIMARY $SYSTEM-B
:PUP PRIMARY $SYSTEM-MB
```

Now, issue the LISTDEV command. Notice that the asterisks are next to $SYSTEM-B and $SYSTEM-MB. Enter:

```
:PUP LISTDEV $SYSTEM
```

The following information is returned:

```
LDEV NAME   STATE PPID    PCU   BPID    BCU  DMNT TYPE SUBTYPE RECSIZE  DPTYPE

4  $SYSTEM-P      00,004  %10   01,004  %10          3     3      4096    DP2
   $SYSTEM-B   *  00,004  %110  01,004  %110
   $SYSTEM-M      00,004  %11   01,004  %11
   $SYSTEM-MB  *  00,004  %111  01,004  %111
```

REBUILDDFS Command

The REBUILDDFS command rebuilds the disc free-space table on a specified disc volume.

The syntax of the REBUILDDFS command is:

```
REBUILDDFS  { $<volume-name> }
            { $<ldev-number> }
```

$<volume-name>

is the name of the volume whose free-space table is to be rebuilt.

$<ldev-number>

is the logical device number of the volume whose free space table is to be rebuilt.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- Use REBUILDDFS under the following circumstances:

  --GUARDIAN file-system error 58 has occurred (DISC FREE SPACE TABLE IS BAD).

  --It is suspected that free space has been lost over a period of time. One way that this might happen is for an I/O error to occur during a PURGE command, causing the system to delete the file label from the directory but fail to deallocate the file extents.

- A version of the PUP REBUILDDFS command prior to A03 cannot be used after one of the following disc operations has been performed in a version later than A03:

  PUP REPLACEBOOT command
  PUP LABEL command
  PUP REBUILDDFS command
  Tape-to-disc cold load

- The REBUILDDFS command always rebuilds the free-space tables on both discs of a mirrored volume (unless one of the two discs is not ready or is in a hard down state because of a hardware problem).

- The REBUILDDFS command can be issued only when the volume is down. The command prevents file opens while it executes.

- The REBUILDDFS command brings the disc into an up state, with opens allowed.

- If two files are found to overlap the extent (an error condition), REBUILDDFS purges one of the files. An error message is generated indicating the file that was purged.

- The free-space table on $SYSTEM can be rebuilt only if the disc is in a down state. The following procedure shows the steps necessary to bring the system disc down so that a PUP REBUILDDFS can be performed.

Rebuilding the Free-Space Table on $SYSTEM

There are always files open on the $SYSTEM volume (for example, USERID and OSIMAGE) when the system is operational. The GUARDIAN command interpreter program file must be transferred to another volume and started so it can be used to run a copy of PUP. In this procedure, you transfer the program files for PUP and COMINT to a volume named $HOLD, down the $SYSTEM volume, and perform a REBUILDDFS.

NOTE

After performing this procedure, you must cold load the system.

CAUTION

This procedure should be viewed as an emergency action. We do not guarantee that it will always work even if the system is quiesced. If this procedure fails, it may leave $SYSTEM in an unusable state. To prepare for recovery, you should first remove the $SYSTEM mirror or be prepared to do a cold load from tape (this destroys all existing files on the $SYSTEM disc).

1. :LOGON SUPER.SUPER
2. :FUP DUP $SYSTEM.SYS<nn>.COMINT, $HOLD.BUILD.COMINT
3. :FUP DUP $SYSTEM.SYS<nn>.PUP, $HOLD.BUILD.PUP
4. :RUN $HOLD.BUILD.COMINT/IN $<term>, OUT $<term>, NOWAIT/
5. :EXIT

Step 4 creates a second COMINT with the RUN command. In step 5, you exit from the command interpreter that was running on $SYSTEM, in effect stopping it.

6. :LOGON SUPER.SUPER

At this point, you have logged onto the command interpreter that is running from the $HOLD volume. Stop any applications that are running on $SYSTEM and notify all users to close any open files (use FUP LISTOPENS to find out what files are still open). All activity must be stopped.

7. :RUN $HOLD.BUILD.PUP
8. :PUP
9. #DOWN all line handlers
10. #REFRESH $SYSTEM, ALL
11. #STOPOPENS $SYSTEM
12. #DOWN ! $SYSTEM
13. #REBUILDDFS $SYSTEM
14. #EXIT

If, when performing the above procedure, you get the message

NO DISC SPACE FOR VIRTUAL MEMORY

you should duplicate the FUP program file from $SYSTEM to another volume and then repeat the procedure from Step 2. This avoids the problem, whether it is due to a corrupted free-space table or insufficient free space on the disc.

Example

To rebuild the free-space table on the volume named $SALES, enter:

```
:PUP REBUILDDFS $SALES
```

REFRESH Command

The REFRESH command writes current file control block (FCB) information to file labels on the disc volume, ensuring that these file labels represent the actual state of the files.

The syntax of the REFRESH command is:

```
REFRESH [ $<volume-name>  ]  [ , ALL ]
        [ $<ldev-number>  ]
```

$<volume-name>

is the name of the volume whose associated file control blocks (FCBs) should be written to disc. If omitted, all FCBs for all volumes are written to their respective discs.

$<ldev-number>

is the logical device number of the volume whose associated FCBs should be written to disc. If omitted, all FCBs for all volumes are written to their respective discs.

ALL

in addition to main memory FCBs, also writes to disc all audit buffers, dirty cache pages, and dirty FCBs.

Considerations

- While a file is open, the file control information is kept in its main-memory-resident FCB; this control information is written to the physical volume only under certain conditions (for instance, when the file is closed, when a REFRESH operation is performed, when a file changes and it has the REFRESH attribute on, or when there have been no disc requests against the file for 30 seconds). The execution of the

REFRESH command writes the information contained in all FCBs to the file labels on the associated disc volume.

- The REFRESH command should be used prior to a tape backup of open disc files in cases where the application is always running. At some point during the day when the system is quiescent (no transactions are taking place), issue a REFRESH command for all volumes in the system. Then, when the files are backed up, the file labels on the backup tape represent the actual states of the files backed up.

- The REFRESH, ALL operation should be performed prior to shutting down the total system or prior to removing a disc pack. This ensures that the file labels on the disc represent the states of the files on that disc.

Examples

1. To write file information from the FCBs to the file labels on the disc named $WORK, enter:

```
:PUP REFRESH $WORK
```

2. Before removing the nonmirrored disc pack $NEW, update the file labels by entering:

```
:PUP REFRESH $NEW, ALL
```

REMOVE Command

The REMOVE command performs an implicit REFRESH, ALL on a volume of a mirrored pair and then logically downs the specified half so that it can be physically removed. Before physically removing one pack of a mirrored volume, issue a REMOVE command to ensure that the data on the pack is consistent with any files open on the volume.

Note that the REMOVE command is not valid for nonmirrored volumes, and is rejected if either device of the mirrored volume is down (inoperable).

The syntax of the REMOVE command is:

```
REMOVE [!] { $<volume-name>  [-P]  }
           {                 [-M]  }  [ , IGNOREBADSECTORS ]
           { $<ldev-number>  [-P]  }
           {                 [-M]  }
```

!

indicates that PUP is to proceed with the REMOVE operation even if files are open on the pack to be removed.

$<volume-name>

is the volume name of the disc pack to be removed.

$<ldev-number>

is the logical device number of the disc pack to be removed.

-P | -M

specifies that the primary or mirror volume is to be removed.

→

IGNOREBADSECTORS

overrides the protection feature that causes REMOVE to terminate if unspared sectors exist in either mirror.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- When the REMOVE command executes, it performs an implicit REFRESH, ALL operation (as described in the discussion of the PUP REFRESH command) for the volume. This ensures that all data on the pack is valid for any files that may be open. Following the refresh, the device where the pack is to be removed is marked as down. This command is typically issued when the removed pack is to be used for volume backup and another pack installed and brought online through a REVIVE command.
- Before issuing a REMOVE command, each application should be brought to some quiescent point so that the state of the files is meaningful. This does not require application programs to close their files but probably does require that the current transaction be completed. Files in EDIT format should be closed.
- Before issuing a REMOVE command, you should determine whether any unspared bad sectors exist on the disc and spare them (using the PUP commands LISTBAD and SPARE). The REMOVE operation terminates abnormally if unspared sectors exist on the disc. To avoid this without sparing bad sectors, specify the IGNOREBADSECTORS option.

Example

To perform a REFRESH operation on the primary disc named $NEW and then down the device on which the volume is mounted, enter:

```
:PUP REMOVE $NEW-P
```

REMOVEAUDITED Command

The REMOVEAUDITED command enters, in the Transaction Monitoring Facility (TMF) catalog, the name of a disc volume to be removed and the names of all the audited files on that volume. Refer to the Transaction Monitoring Facility (TMF) Reference Manual for more information about TMF.

The REMOVEAUDITED command is used (in place of the REMOVE command) before physically removing one pack of an audited mirrored volume for a TMF online dump. This command is typically used when the removed pack is to be used for volume backup and when another pack is mounted and brought online through a REVIVE command.

NOTE

Because of the format of the DP2 volume label, you cannot use the REMOVEAUDITED command on a DP2 volume. If you do, you get an error message (DISC ON-LINE DUMPS ARE NOT AVAILABLE FOR DP2 DISC VOLUMES.)

The syntax of the REMOVEAUDITED command is:

```
REMOVEAUDITED  [ / OUT <listfile> / ] { $<volume-name> [-P] }
                                      { $<volume-name> [-M] }
                                      { $<ldev-number> [-P] }
                                      { $<ldev-number> [-M] }

             , PACK <pack> [ , LIST ] [ , IGNOREBADSECTORS ]
```

<listfile>

is the output device for the listing.

$<volume-name>

is the volume name of the disc pack to be removed.

→

$<ldev-number>

is the logical device number of the disc pack to be removed.

-P | -M

indicates that the primary or mirror disc is to be removed.

PACK <pack>

specifies the identifier by which the physical disc pack is known to the TMF catalog. <pack> is a six-character alphanumeric identifier encoded on the disc pack when REMOVEAUDITED creates an online dump. Note that the <pack> identifier names the physical disc but not the data recorded on it or the volume name of the disc drive. The identifier is erased from the volume label if the pack is rolled forward or if the FORMAT or LABEL command is used.

LIST

lists the files that have been entered in the TMF catalog for this volume. LIST produces a listing similar to that of the TMFCOM DUMP FILES DETAIL command. If LIST is omitted, a listing similar to that of the TMFCOM DUMP FILES BRIEF command is produced. (Refer to the Transaction Monitoring Facility (TMF) Reference Manual for information about TMF commands.)

IGNOREBADSECTORS

overrides the protection feature that causes the REMOVEAUDITED command to terminate if unspared sectors exist in either mirror.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- Before issuing a REMOVEAUDITED command, you should determine whether any unspared bad sectors exist on the disc and spare them (using the PUP commands LISTBAD and SPARE). The REMOVEAUDITED operation terminates abnormally if unspared sectors exist on the disc. To avoid this without sparing bad sectors, specify the IGNOREBADSECTORS option.
- After the REMOVEAUDITED operation, a REFRESH operation is performed for the volume, and the volume is marked as down.
- If you use the REMOVE (instead of the REMOVEAUDITED) command on an audited disc, of if you use the REMOVEAUDITED (instead of the the REMOVE) command on a nonaudited disc, the REMOVE operation proceeds, but the following warning is printed upon completion:

  THE REMOVE WAS SUCCESSFUL BUT IT IS NOT A TMF DUMP.

- If the disc has any defective sectors, you must use the PUP command SPARE before using the REMOVEAUDITED command on that disc.
- The REMOVEAUDITED command proceeds even if there are open files on the disc.
- The REMOVEAUDITED command fails if the pack contains audit trails or if it is not mirrored.
- The REMOVEAUDITED command functions normally in a TMF installation configured with the TMFCOM ALTER FILES, RECOVERY OFF, but the resulting dump can never be used.

Example

To enter the volume name $STORE-M and the names of all audited files on $STORE-M in the TMF catalog, to write the pack identifier PROG01 to the volume label, to refresh $STORE-M, and to down the device on which $STORE-M is mounted, enter:

```
:PUP REMOVEAUDITED $STORE-M, PACK PROG01
```

RENAME Command

The RENAME command provides a convenient method for changing the name of a demountable disc volume without destroying the data on the volume. This command replaces the MOUNT command once used for NonStop systems. Use of PUP MOUNT is no longer allowed.

The syntax of the RENAME command is:

```
RENAME [!] { $<old-volume-name> } , $<new-volume-name>
           { $<ldev-number>     }

       [ , PERMANENT ]
```

!

specifies that the RENAME operation is to proceed even if the new name is being used by another device or process. If the new name is already being used, the volume is left demounted.

$<old-volume-name>

is the volume name of the disc to be renamed.

$<ldev-number>

is the logical device number of the disc to be renamed.

$<new-volume-name>

is the new volume name to be given to the disc volume.

PERMANENT

specifies that both the original and alternate names of the disc are assigned the new name. If this parameter is not specified, only the alternate name is changed.

Considerations for All Disc Volumes

- All disc volumes have two names: an original name and an alternate name. Both are assigned the same name when a LABEL is performed.
- When a system is cold loaded or when the PUP UP command is used on a demounted volume, the original name is used to bring up the device, unless this name is being used by some other device or process. If the original name is in use, then the alternate name for the volume is used.
- The entire device must be initially in the down state or the special request state.
- Each disc renamed must have a valid volume label.
- When a volume is renamed, its timestamp is updated.
- After the RENAME operation, implicit DEMOUNT and UP operations are performed on the device. The name of the volume that is brought up depends on whether or not the original name is in use.

Considerations for Mirrored Discs

- If you specify $<old-volume-name>, the alternate and original names of both mirrors of the volume are checked to verify that they have the same name as $<old-volume-name>, and:

  --If both mirrors have $<old-volume-name> as either their original name or their alternate name, then both mirrors are renamed.

  --If only one mirror has $<old-volume-name> as its original name or alternate name, then just that one is renamed, and the following message appears for the mirror that has a different name:

```
WARNING: $<ldev-num> { -P } CONTAINS THE VOLUME $<vol-name>
                     { -M }
```

--If neither mirror has $<old-volume-name> as either its original name or its alternate name, then neither is renamed, and the following messages, one for each disc, appears:

```
WARNING: $<ldev-number> -P CONTAINS THE VOLUME $<vol-name>
WARNING: $<ldev-number> -M CONTAINS THE VOLUME $<vol-name>
```

- If you specify $<ldev-number>, the discs are not checked to verify that they have the same name as $<old-volume-name>.

- If the timestamps are different on a mirrored pair, only the newer disc are renamed. A message is generated for the disc that is not renamed:

```
WARNING: $<ldev-num> { -P } IS NOT CONSISTENT WITH ITS MIRROR
                     { -M }
```

Considerations for Nonmirrored Discs

- If you specify $<old-volume-name>, the original and alternate names are checked to verify that they have the same name as $<old-volume-name>. At least one name must match $<old-volume-name>, or the following message is displayed and the RENAME operation stops:

```
WARNING: $<ldev-number> CONTAINS THE VOLUME $<volume-name>
```

- If you specify $<ldev-number>, the disc is renamed regardless of whether the original or alternate name matches the $<old-volume-name>.

Considerations for the ! parameter

- If you specify ! with the PERMANENT option and the new name is being used by a device or another process, the disc is renamed but left demounted.

- If you specify ! without the PERMANENT option and the new name is being used by a device or another process, just the alternate name gets assigned the $<new-volume-name>. If the original name is also in use, the volume is left demounted. If the original name is not in use, the volume comes up under the original name.

- If you specify ! (with or without the PERMANENT option) and the new name is being used by a device or another process, the following error message is displayed:

  THE NAME $<new-volume-name> IS BEING USED BY SOME OTHER DEVICE OR PROCESS

Examples

1. To illustrate the results of the RENAME command, first enter the LISTDEV command.

   :PUP LISTDEV DISC

   The following information is displayed:

```
LDEV NAME STATE PPID    PCU   BPID    BCU  DMNT  TYPE  SUBTYPE RECSIZE DPTYPE

4  $SYSTEM-P *  00,004  %10   01,004  %10          3       2       4096   DP2
   $SYSTEM-B    00,004  %110  01,004  %110
   $SYSTEM-M *  00,004  %11   01,004  %11
   $SYSTEM-MB   00,004  %111  01,004  %111
5  $DATA-P    D 00,005  %114  01,005  %114  Y     3       3       4096   DP1
   $DATA-B    D 00,005  %14   01,005  %14
```

   Notice that the $DATA volume is in the down state. Now enter the RENAME command to rename $DATA:

   :PUP RENAME $DATA, $NEW, PERMANENT

   To show that $DATA has been renamed, enter the LISTDEV command:

   :PUP LISTDEV DISC

   The following information is displayed:

```
LDEV NAME STATE PPID    PCU   BPID    BCU  DMNT  TYPE  SUBTYPE RECSIZE DPTYPE

4  $SYSTEM-P *  00,004  %10   01,004  %10          3       2       4096   DP2
   $SYSTEM-B    00,004  %110  01,004  %110
   $SYSTEM-M *  00,004  %11   01,004  %11
   $SYSTEM-MB   00,004  %111  01,004  %111
5  $NEW-P     * 00,005  %114  01,005  %114  Y     3       3       4096   DP1
   $NEW-B       00,005  %14   01,005  %14
```

2. Assume in this example that you accidentally purged a file on the volume $DATA, and the file that you purged is backed up on an older version of $DATA. The first thing that must be done is to down $DATA-M and $DATA-MB (using the REMOVE command).

```
:PUP
#REMOVE $DATA-M
#LISTDEV DISC
```

The listing below shows this state:

```
LDEV NAME STATE PPID    PCU   BPID    BCU   DMNT  TYPE  SUBTYPE RECSIZE  DPTYPE

6  ? -P      D  00,003  %114  01,003  %114   Y      3       3      4096    DP1
   ? -B      D  00,003  %14   01,003  %14
7  $DATA-P   *  00,004  %10   01,004  %10    Y      3       3      4096    DP1
   $DATA-B      00,004  %110  01,004  %110
   $DATA-M   D  00,004  %11   01,004  %11
   $DATA-MB  D  00,004  %111  01,004  %111
```

Now physically replace the unit plug for $7-M with a unit plug number 4, physically replace the disc $DATA-M with the archived copy of $DATA, and then rename the disc ($TEMP).

```
#RENAME $6, $TEMP
```

Note that a LISTDEV still displays the permanent name of "?" because the RENAME command changes only the alternate name of disc $6:

```
#LISTDEV DISC
```

The following information is displayed:

```
LDEV NAME STATE PPID    PCU   BPID    BCU   DMNT  TYPE  SUBTYPE RECSIZE  DPTYPE

6  ? -P         00,003  %114  01,003  %114   Y      3       3      4096    DP1
   ? -B         00,003  %14   01,003  %14
7  $DATA-P   *  00,004  %10   01,004  %10    Y      3       3      4096    DP1
   $DATA-B      00,004  %110  01,004  %110
   $DATA-M   D  00,004  %11   01,004  %11
   $DATA-MB  D  00,004  %111  01,004  %111
```

Then FUP DUP the file that you purged, and do the following:

```
#DOWN $TEMP
#DEMOUNT $TEMP
```

Replace unit plug 4 with unit plug 1, and replace the archive copy of $DATA with the original $DATA-M.

```
#REVIVE $DATA
#EXIT
```

REPLACEBOOT Command

The REPLACEBOOT command allows the replacement of the microcode and the disc cold-load bootstrap programs without performing a tape-to-disc cold load.

The REPLACEBOOT command can be used only with versions of the GUARDIAN operating system available since software release A03 (April 1982).

The syntax of the REPLACEBOOT command is:

```
REPLACEBOOT  { $<volume-name> } , <filename>
             { $<ldev-number> }
```

$<volume-name>

is the name of the volume whose bootstraps and microcode are to be replaced. The disc must be in the up state before entering the command. (If it is mirrored, then only those halves that are up are changed.)

$<ldev-number>

is the logical device number of the disc whose bootstraps and microcode are to be replaced. The disc must be in the up state before entering the command. (If it is mirrored, then only those halves that are up are changed.)

<filename>

specifies the name of a file to which the SYSGEN system image tape file, ZSYSDISC.SYSDISC, is copied. The system volume and subvolume name can be changed by the RESTORE program, but the file name "SYSDISC" is restored under the same name. (Refer to Section 4 for more information about RESTORE.)

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)

- Every system disc has microcode sections; for example, bootstrap microcode, 3107 microcode, and so forth. These microcode sections are updated and replaced on the disc by the REPLACEBOOT command or when a cold load from tape to disc is performed.

NOTE

The 3107 downloadable controller is available only for NonStop TXP processors.

- If an A03 or later version of one of the following operations has been performed on a disc:

  PUP REPLACEBOOT command
  PUP LABEL commmand
  PUP REBUILDDFS command
  Tape-to-disc cold load

  then an earlier version (prior to A03) of the PUP REBUILDDFS command cannot be used on that disc.

- When replacing disc controller microcode and disc cold-load bootstraps, if the specified disc does not contain space for the cold-load bootstraps or if the currently allocated space is insufficient for the new bootstraps, then the REPLACEBOOT command allocates the appropriate space on the disc. It may be necessary for the user to compress the disc space (using DCOM, described in Section 8) or delete extraneous files to provide sufficient space for the new bootstraps.

- If it is necessary for PUP to allocate additional space for the new bootstraps, the file ZSYSDISC.PUPDSC is created temporarily on the volume. If a file with this name already exists, the REPLACEBOOT operation fails.

- If the subtype of the volume is different from the subtype specified when the system image was produced, an error message is printed.

- If an I/O failure or system crash occurs during the REPLACEBOOT operation, the disc bootstrap may no longer be valid; in this case, it is not possible to cold load the system from $<disc-device>.

- If an error occurs, PUP indicates whether or not the disc bootstrap is still usable by displaying the message:

```
THE DISC'S BOOTSTRAP IS { STILL    } USABLE
                        { NO LONGER }
```

- To facilitate recovery in the event that an I/O failure occurs, back up all files on the volume by performing a PUP REMOVE before using the REPLACEBOOT command. If the volume is not mirrored, use the BACKUP program.

- If a system crash occurs during the REPLACEBOOT operation and the system is running from the mirrored volume specified in the REPLACEBOOT command, then the backup disc (created as described above) can be used to cold load the system.

- If a system crash occurs during the REPLACEBOOT operation, and the volume is not mirrored and no other system disc exists, you can perform a tape-to-disc cold load, and use the BACKUP tape (created as described above) to restore the volume files.

- If an operating system image is restored to a disc that has been labeled with an A03 or later version of the PUP LABEL command, the REPLACEBOOT command can be used to create the disc bootstraps and microcode. Therefore, it is no longer necessary for the PUP LABEL command to reserve space for bootstraps and microcode on discs that do not contain system images.

REVIVE Command

The REVIVE command is used to bring an inoperable device of a mirrored disc volume back into operation. The REVIVE command copies data from the operable device to the down device while concurrent processing takes place. The REVIVE command is also used to perform disc volume backup.

NOTE

The pack on the device being revived must be formatted before the REVIVE operation can be performed. (A formatted scratch pack containing data or a previous pack from the same disc drive may be used if desired.)

The syntax of the REVIVE command is:

```
REVIVE  { $<volume-name> }
        { $<ldev-number> }

        [ , <num-tracks> [ , <copy-interval> ] ]
```

$<volume-name>

is the name of the mirrored volume being revived.

$<ldev-number>

is the logical device number of the mirrored volume being revived.

<num-tracks>

is an integer indicating the number of tracks to be copied in each copy interval. If omitted, one track is copied in each interval.

→

<copy-interval>

is an integer indicating the maximum interval, in 10-millisecond units, between each copy operation and the next. The default setting is 100 10-millisecond units (that is, 1 second). (See "Considerations.")

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- The REVIVE command brings an inoperable device of a mirrored volume back into operation by copying data from the operable device to the down or removed device. Groups of tracks are copied from the operable device to the device being revived at periodic intervals. The number of tracks in a copy group and the time period between copying groups can be specified. This permits the copy to take place with a minimum of system throughput degradation.
- A disc must be in the down state for a successful REVIVE. If the disc is in the hard down state, you must perform a PUP DOWN to put it in the down state. If you try to revive a hard-down disc, you get GUARDIAN file-system error 66 (DEVICE HAS BEEN DOWNED BY OPERATOR).
- When you enter the REVIVE command, PUP initiates the REVIVE operation, and the disc-process portion of the operating system completes it. As a result, once the operation has been initiated (but not necessarily completed), PUP prompts for another command.
- The parameters <copy-interval> and <num-tracks> together determine the rate at which the REVIVE proceeds. The REVIVE proceeds in the following manner:

  1. A number of tracks equal to <num-tracks> are copied from the primary to the mirror, or the mirror to the primary.
  2. The disc process checks for other work to be done; if there is none (no other requests pending), step 1 is repeated.

3. If there are other requests, the disc process temporarily delays the REVIVE operation by a time interval of (<copy-interval> * 10) milliseconds and then proceeds to step 4.
4. The disc process begins processing other requests.
5. When the specified delay interval expires, the disc process completes the single request that it is currently processing and then returns to step 1.
6. When there are no more tracks to be copied, the REVIVE operation is complete. At that time, the revived disc device is automatically reintroduced (with the UP command).

- You can change the REVIVE rate by reentering the REVIVE command and specifying different values for <num-tracks> and <copy-interval>. The disc process does not start the REVIVE from the beginning but continues from where it was stopped with the new values.

- To display the status of a REVIVE operation, use the LISTDEV command with the DETAIL option. (See the LISTDEV command for additional information about this option.)

- You can stop the REVIVE operation either by issuing a PUP DOWN command on the part of the device that is in the revive state, or by stopping the process ID.

- If a nonfatal error (such as a checksum error) is encountered, the REVIVE does not proceed, but continues to retry at the current address until the operation is successful or the REVIVE is suspended or stopped. With each retry, the <copy-interval> doubled after each retry so that the retrying does not consume system resources or produce too many console messages. The <copy-interval> continues to double up to a maximum of one hour; at this level it retries indefinitely. Once the REVIVE succeeds, <copy-interval> returns to its original value.

- A stalled REVIVE indicates a media problem requiring operator action. Stop the REVIVE operation and examine the console messages generated by the nonfatal error, referring to the System Messages Manual for information on how to recover from the error.

- If both devices of a mirrored volume are inoperable, the REVIVE command does not work. In this case, you must use the UP command to place one of the discs back into operation.

- If both devices of a mirrored volume are inoperable and you have previously taken a backup (with the REMOVE command), you can use the REVIVE command to bring the entire mirror volume back into service. This procedure is described under "Backing

Up and Restoring System Files" in the System Operator's Guide.

- If both devices of a mirrored volume are inoperable, and you have previously taken a backup (with the PUP REMOVE command), you can use the REVIVE command to bring the entire mirror volume back into service. This procedure is described under "Backing Up and Restoring System Files" in the System Operator's Guide.

Example

To copy 20 tracks at a time at 100-millisecond intervals from the operable volume to the inoperable volume of the mirrored pair named $STORE1, enter:

```
:PUP REVIVE $STORE1, 20, 10
```

SETCACHE Command

The SETCACHE command permanently alters the cache configuration (set initially by the PUP LABEL command) for an in-use volume. This command is available for DP2 discs only.

CAUTION

Using this command causes all disc caches to be flushed when the cache configuration is changed. Overly restrictive or extravagant cache configurations can cause severe performance problems.

The syntax of the SETCACHE command is:

```
SETCACHE  { $<volume-name> }
          { $<ldev-number> }

          [ , [ <block-size> = <num-blocks> ] ... ]
```

$<volume-name>

is the name of the volume whose cache configuration is to be altered.

$<ldev-number>

is the logical device number of the volume whose cache configuration is to be altered.

<block-size> = <num-blocks>

specifies a new volume cache configuration in the form of a string of pairs, where <block-size> is a particular cache size, and <num-blocks> is the value of the desired allocation. See "Examples."

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)

- Disc cache configuration is the number and size of sectors read from disc and stored in the processor for use as virtual memory.

- Any cache not specified with this command retains its prior allocation (set by default or with the PUP LABEL command).

- A valid DP2 <block-size> for a given controller is a power-of-two multiple of the sector size, up to the maximum transfer size. <block-size> can be:

512 bytes (sector size)	or	.5K
1024	or	1K
2048	or	2K
4096 (maximum transfer size)	or	4K

  If your cache configuration does not allow at least two blocks for each cache on each disc process in the volume disc-process group, the disc process overrides your requested values with the minimum number of blocks required. This is done to ensure that the cache is adequate to satisfy any request requiring a given size buffer.

- To avoid wasting unused page fragments, all allocations round up to the next whole page (for example, 512 = 6 becomes 512 = 8, or 2 pages). While the pages underlying the cache are not swappable, they are available to the memory manager when not in use.

- The maximum number of cache buffers that can be allocated for each cache is 8192. The minimum number is one.

- If user-requested allocations exceed the physical memory that can be used by the disc process, the allocations are trimmed back until they fit. The amount of physical memory that can be used by the disc process depends on the installation, but in no case can it be more than 10 megabytes.

Example

The following SETCACHE command sets up the cache configuration shown by the LISTCACHE example that follows:

```
:PUP SETCACHE $DP2, 512=100, 2K=100, 4K=100
:PUP LISTCACHE $DP2
```

The following information is returned:

```
CACHE CONFIGURATION: \SYSB.$DP2        TIME: 15 JAN 1985, 16:30:10

CACHE BLOCK SIZE:     512          1024          2048          4096

BLOCKS REQUESTED:     100                        100           100
BLOCKS ALLOCATED:     100             4          100           100

BYTES ALLOCATED TO CACHE:    00667 KB          WRITES/CONTROL POINT:  0000
```

SPARE Command

The SPARE command assigns an alternate sector from the available spare sectors on a specified disc volume for use in place of a sector indicated as (defective) bad by the LISTBAD command. It enters that sector into the lifetime defect log for the disc.

CAUTIONS

Sectors should not be spared while processing is taking place because inconsistent data might be copied to an alternate sector.

Do not attempt to spare a disc that already has a SPARE operation in progress. Only one SPARE should be performed at a time.

The syntax of the SPARE command is:

```
SPARE { $<volume-name> [ -P ] }
      { $<ldev-number> [ -M ] }

    , <cylinder> , <head> , <sector> [ , PHYS ]
```

$<volume-name>

is the name of the volume where a bad sector is to be assigned an alternate sector.

$<ldev-number>

is the logical device number of the volume where a bad sector is to be assigned an alternate sector.

→

-P | -M

indicates that the SPARE command is for the primary or mirror volume. -P or -M must be specified if the volume is a mirrored volume.

<cylinder>

is the cylinder number on $<volume-name> or $<ldev-number>.

<head>

is the number of the head on $<volume-name> or $<ldev-number>.

<sector>

is the sector to be assigned an alternate sector.

PHYS

if present, indicates that the specified address is a physical sector address. If omitted, the address is interpreted as a logical sector address.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- After a bad sector is assigned an alternate, it is no longer listed when a LISTBAD command is executed.
- When a sector on a nonmirrored volume is spared, data from the bad sector is copied to the new sector. The data from the bad sector may be invalid.

- When a sector on half of a mirrored volume is spared, the sector data is copied from the other half into the new sector. The new sector therefore contains valid data. If half of a mirrored volume is down, sectors are copied into alternate sectors just as they are on a nonmirrored volume.

- Your PUP SPARE command must be in the form in which the system supplied the address. Most console messages report the address in a logical form. Exceptions to this are console messages 97 and 98 which use the PHYS form, as noted in the console message. You must specify either the PHYS option for a physical byte address or the default (omit the PHYS option) for a logical byte address.

- PUP LISTDEFECTS displays all defective sectors, and PUP LISTLOG displays those sectors already entered into the defect log. Use PUP SPARE with the PHYS option to spare the physical sectors that do not appear in the defect log. A subsequent LISTLOG command can verify whether the correct sectors have been spared.

Example

To assign an alternate sector to be used in place of the sector with cylinder number 173, head number 2, and sector number 6 on the mirror device of the mirrored volume named $DATA, enter:

```
:PUP SPARE $DATA-M, 173, 2, 6
```

STATUS Command

The STATUS command displays controller identification information about a downloadable controller and, for the 3206 Tri-Density Tape Controller, displays its idle loop and failure history.

The syntax of the STATUS command is:

```
STATUS  [ / OUT <listfile> / ]

        CONTROLLER [%] <cpu> , [%] <controller>
```

<listfile>

specifies a disc file, nondisc device, or process to which PUP directs its listing output. If omitted, the listing is sent to your home terminal.

CONTROLLER [%] <cpu> , [%] <controller>

<cpu>

is a CPU number identifying one of the processors to which the downloadable controller is attached. If this parameter is preceded by a percent sign (%), PUP treats it as an octal number.

<controller>

is the number of the controller. If this parameter is preceded by a percent sign (%), PUP treats it as an octal number.

Considerations

- This command words for downloadable controllers only. If the controller microcode is readable only (cannot be written to) then PUP returns the message:

  NO STATUS IS AVAILABLE FOR THIS DEVICE

- You can use the PUP STATUS command after receiving console message 100 (MICROCODE LOADING FAILURE) indicating that the LOADMICROCODE command cannot find the correct microcode file to download. The LOADMICROCODE command searches for the file named:

  $SYSTEM.M<hpn>.<hhhppp>

  <hpn> is the highest assembly part number of the controller board to be loaded.

  <hhh> is the hardware revision number of the board.

  <ppp> is the PROM revision number.

- The PUP STATUS command returns <hpn>, <hhh> and <ppp> as part of its output. Use these numbers to construct the file name of the microcode file. Then use the RESTORE program (described in Section 4 of this manual) to restore the correct microcode file from the site update tape.

Listing Format

The format of the STATUS display for all downloadable controllers except the 3206 Tri-Density Tape Controller is shown in Figure 5-22.

```
CONTROLLER HIGHEST ASSEMBLY PART NUMBER:  <hpn>
CONTROLLER SERIAL NUMBER:  <nnn>
PROM PART NUMBER:  <prom-pn>
PROM REVISION LEVEL:  <ppp>
HARDWARE REVISION LEVEL:  <hhh>
```

Figure 5-22. STATUS Listing Format for All Downloadable Controllers Except the 3206 Tri-Density Tape Controller

Listing headers are identified as follows:

CONTROLLER HIGHEST ASSEMBLY PART NUMBER	is the highest assembly part number <hpn> of the controller board.

CONTROLLER SERIAL NUMBER	is the serial number <nnn> of the controller board.
PROM PART NUMBER	is the part number <prom-pn> of the PROM containing the controller resident microcode.
PROM REVISION LEVEL	is the revision level number <ppp> of the PROM containing the controller-resident microcode.
HARDWARE REVISION LEVEL	is the revision level number <hhh> of the controller hardware.

The format of the STATUS display for the 3206 Tri-Density Tape Controller is shown in Figure 5-23.

```
CONTROLLER HIGHEST ASSEMBLY PART NUMBER:  <hpn>
PROM PART NUMBER:  <prom-pn>
PROM REVISION LEVEL:  <ppp>
HARDWARE REVISION LEVEL:  <hhh>
MICROPROGRAM REVISION:  <mm>

  CONTROLLER ERROR STATISTICS.

IDLE LOOP FAILURE HISTORY.  NUMBER OF ITERATIONS:    [0] <iterations>
    TEST          FAILURES       STATUS

    PROCESSOR   [(0)] <fn>    <failure-type>
    PARITY      [(0)] <fn>
    R/W LOOP    [(0)] <fn>
    REGISTER    [(0)] <fn>
    BUFFER      [(0)] <fn>
    CTC         [(0)] <fn>

I/O FAILURE HISTORY.
             READ ERRORS      WRITE ERRORS

UNIT <tn>   [(0)]  <rn>       [(0)]  <wn>
UNIT <tn>   [(0)]  <rn>       [(0)]  <wn>
UNIT <tn>   [(0)]  <rn>       [(0)]  <wn>
UNIT <tn>   [(0)]  <rn>       [(0)]  <wn>
```

Figure 5-23. STATUS Listing Format for the Tri-Density Tape Controller

Listing headers are identified as follows:

CONTROLLER HIGHEST ASSEMBLY PART NUMBER	is the highest assembly part number <hpn> of the controller board.
PROM PART NUMBER	is the part number <prom-pn> of the PROM containing the controller-resident microcode.
PROM REVISION LEVEL	is the revision level number <ppp> of the PROM containing the controller-resident microcode.
HARDWARE REVISION LEVEL	is the revision level number <hhh> of the controller hardware.
MICROPROGRAM REVISION	is the revision number <mm> of the downloaded microcode that drives the controller.
NUMBER OF ITERATIONS	indicates the number of test iterations performed by the controller idle loop.
TEST FAILURES STATUS	indicate the type of test conducted (for instance, PROCESSOR or PARITY check) by the controller self-testing mechanism during the controller idle loop; the number of failures <fn> for each test type; and the type of failure, <failure-type>. If present, (O) indicates a failure-count overflow.
READ ERRORS WRITE ERRORS	indicate the number of read and write errors, <rn> and <wn> respectively, detected by the controller for each tape unit <tn>. This may differ from the number of errors detected by the application program due to successful error recovery by the controller. If the number of retry errors a day is significantly high, maintenance is probably required--for instance, the drive could be dirty or the medium could be bad. If present, (O) indicates an error-count overflow.

Examples

1. To display status information for controller 2, a Tri-Density Tape Controller attached to CPU 6, enter:

```
:PUP STATUS CONTROLLER 6, 2
```

The following information is returned:

```
CONTROLLER HIGHEST ASSEMBLY PART NUMBER: 56250
PROM PART NUMBER: 58295
PROM REVISION LEVEL: A00
HARDWARE REVISION LEVEL: A00
MICROPROGRAM REVISION: A0

  CONTROLLER ERROR STATISTICS.

IDLE LOOP FAILURE HISTORY.   NUMBER OF ITERATIONS:   538976288
    TEST         FAILURES    STATUS

    PROCESSOR  (0)     10    HARD FAILURE
    PARITY              4
    R/W LOOP            1
    REGISTER            0
    BUFFER          32767
    CTC                 0

I/O FAILURE HISTORY.
          READ ERRORS    WRITE ERRORS

UNIT 0        22774      (0)  22774
UNIT 1            0            0
UNIT 2        19284           19284
UNIT 3        17696           17696
```

2. To display status information for controller 5, a 3107 downloadable disc controller attached to CPU 6 enter:

```
:PUP STATUS CONTROLLER 6, 5
```

The following information is returned:

```
CONTROLLER HIGHEST ASSEMBLY PART NUMBER:  045568
CONTROLLER SERIAL NUMBER:  003456
PROM PART NUMBER:  098765
PROM REVISION LEVEL: A06
HARDWARE REVISION LEVEL: A06
```

STOPOPENS Command

The STOPOPENS command causes any further calls to the file-system OPEN procedure associated with the specified disc volume to be rejected with a GUARDIAN file-system error 61 (NO MORE FILE OPENS PERMITTED).

CAUTION

If you execute a STOPOPENS command on $SYSTEM, you should immediately execute an ALLOWOPENS, SUPERONLY command. Otherwise, if you exit PUP you cannot run it again, and your system disc is inaccessible to any new file opens.

The syntax of the STOPOPENS command is:

```
STOPOPENS { $<volume-name>  }
          { $<ldev-number>  }
```

$<volume-name>

is the name of the volume on which open requests are to be rejected.

$<ldev-number>

is the logical device number of the volume on which open requests are to be rejected.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)

- When the STOPOPENS command is given, PUP responds with:

  <number> FILES OPEN

  STOPOPENS can be issued repeatedly to determine the number of files remaining open (even if files are open, the command stops further opens).

Example

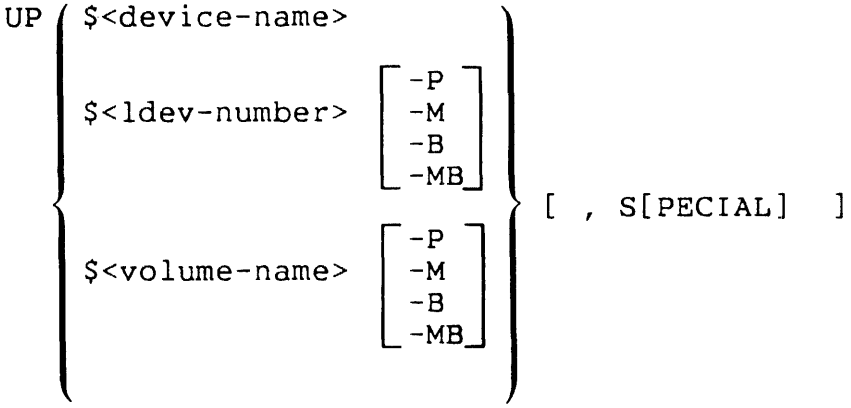
To disallow file-system opens on the volume named $ADMIN, enter:

```
:PUP STOPOPENS $ADMIN
```

UP Command

The UP command is used to make a down device accessible to user processes.

The syntax of the UP command is:



$<device-name>

is the name of the device to be made accessible. $<device-name> is used to bring up a nondisc device.

$<ldev-number>

is the logical device number of the disc volume to be made accessible. $<ldev-number> is used to bring up a nonmirrored disc or both discs of a mirrored disc.

$<volume-name>

is the name of the disc device to be made accessible. $<volume-name> is used to bring up a nonmirrored volume or both volumes of a mirrored volume.

NOTE

The designations -P, -M, -B, -MB apply only to dual-port discs configured as mirrored volumes.

→

-P

places the primary controller path to the primary device up.

-M

places the primary controller path to the mirror device up.

-B

places the backup controller path to the primary device up.

-MB

places the backup controller path to the mirror device up.

S[PECIAL]

sets the logical disc in the special request state so that only privileged processes can access the disc. The -B and -MB designations are not permitted with the SPECIAL option.

Considerations

- This command can be performed only by a user logged on with super-group capability. (Refer to the GUARDIAN Operating System User's Guide for additional information about user IDs.)
- Use the UP command to bring up a disc pack when:

  --The system is already cold loaded.

  --The volume name of the new disc pack is not currently being used by some other device or process.

- To clear the special request state, issue a PUP DOWN command on the part of the device you want cleared.

- When you use the UP command to bring up a disc pack, issue the LISTDEV command to verify that the disc drive actually contains the new pack.

- If you receive a DUPLICATE VOLUME message, you must use the PUP RENAME command rather than the UP command to rename and bring up the new disc pack.

- The UP command is not allowed for a device of a mirrored volume when the other device of the mirrored volume is already up. To make the down device of a mirrored volume operable in such an instance, use the REVIVE command.

- The UP command does not work on hard down paths (that is, paths that have been marked down because of hardware failure). A hard down state is recorded as an "H" in the state field of the LISTDEV display (refer to the description of the LISTDEV command earlier in this section). To bring a controller path out of a hard down state, execute another DOWN command for the device, omitting the HARD option. In issuing this DOWN command, you must be certain to specify the controller path with its identifying tag (-P, -M, -B, or -MB). Any attempt to access a path in a hard down state fails with GUARDIAN file-system error 66 (DEVICE HAS BEEN DOWNED BY OPERATOR). (Refer to the System Messages Manual for additional information about GUARDIAN file-system errors.)

- There are a variety of circumstances under which the disc process decides to down a disc that is being brought up. An error message of some type is printed if:

  --The volume label cannot be read.

  --The controller microcode or unit literal version is not compatible with the driver.

  --The drive is not compatible with the configured subtype.

  --The drive failed during a REBUILDDFS operation, and the REBUILDDFS must be repeated.

  Under the following circumstances however, no console message (other than the down message) is printed:

  --The disc has an older volume label timestamp than its mirror.

  --The disc has track-sparing (not sector-skipping) format.

--The configured disc subtype is incompatible with the subtype specified by the volume label.

--The volume label or directory label format is invalid.

Example

To place back in operation the device named $PRINTER, enter:

```
:PUP UP $PRINTER
```

UPDATEREV Command

The UPDATEREV command is designed for use by Tandem customer engineers and is therefore not described in this manual. For more information about this command, contact your Tandem representative.

SECTION 6

PERUSE

PERUSE is an interactive program that allows you to examine and change the attributes of a spooled job as well as to examine your job while it is in the spooler system. With PERUSE you can:

- Examine a job (for example, a lengthy compiler listing) before deciding whether to print it
- Display a job while it is being spooled
- Monitor changes in the status of a job
- Alter job attributes, such as location, number of copies, or report name
- Print out specified pages rather than the entire file of a job that has been spooled

To perform these functions, PERUSE communicates with the spooler supervisor process and accesses the spooler disc files.

This section contains the complete syntax, considerations, and examples for all the PERUSE commands. Refer to the System Messages Manual for PERUSE error messages. Information on how to use PERUSE and the kinds of tasks that can be accomplished with it are described in the GUARDIAN Operating System User's Guide.

HOW TO RUN PERUSE

You can run PERUSE with the following GUARDIAN command interpreter command:

```
PERUSE  [ / <run-option> / ] [ <supervisor> ]
```

<run-option>

is one or more standard RUN options, separated by commas. The options are described in Section 2 under the GUARDIAN command interpreter RUN command.

<supervisor>

is the name of the spooler supervisor that PERUSE communicates with. If <supervisor> is omitted, then PERUSE assumes $SPLS is the supervisor.

When run, PERUSE sends the following startup message to the terminal:

```
PERUSE  -  T9101B00  -  (18MAR85)     SYSTEM  \EAST
```

T9101B00 is the version of PERUSE.

(18MAR85) is the release date for this version of PERUSE.

\EAST is the system it is running on.

If you have any jobs in the spooler system when PERUSE begins execution, it lists them using the format shown below:

```
JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION   REPORT
456   PRINT  16     1       4          #LPP       ACCOUNT BARB
555   OPEN          1       4    B     #DEFAULT   ACCOUNT BARB
1435  READY  30     1       4     A    #LPRMT3    ACCOUNT BARB
```

Each line in the display stands for a different job. The meanings for each column are:

JOB — shows the job number of each job, as assigned by the supervisor.

STATE — shows the status of each job:

- OPEN — Job is still being collected by the spooler
- READY — Spooler has finished collecting; job is queued and waiting to print
- HOLD — Job has been placed on hold with the HOLD command
- PRINT — Job is currently printing

PAGES — shows the number of pages in each job (OPEN jobs are still being collected, so the number of pages is not known).

COPIES — shows the number of copies to be made of the job.

PRI — shows the priority of the job. When a job is first sent to the spooler, its default priority is 4. The range is 0-7, 0 being lowest.

HOLD — shows the hold status of the job:

- A — Hold-after-printing flag is on
- B — Hold flag is on but the job cannot be placed in HOLD (the job is OPEN)
- X — For some reason the job is in error (for example, TGAL abended while spooling the job)

LOCATION — shows the location of the job.

REPORT — shows the report name that is printed in the job header message.

The display described above can be produced at any time during a PERUSE session using the JOB command.

Displaying a Job

There are three ways to display a job:

- Use the LIST command, which is described later in this section.
- Type a carriage return, which causes PERUSE to list the current line of the current job and increment the position counter. Therefore, five consecutive carriage returns display five consecutive lines.
- If you have a page-mode terminal, use the function keys to list lines from the current job. The higher the number of the function key, the more lines it displays. You can see two lines with <F1>, four lines with <F2>, eight with <F3>, and so forth; <F5> shows more than a page. This geometric progression is shown in Table 6-1, which lists the number of lines displayed by each function key.

Table 6-1. Number of Lines Listed by the Function Keys

Function Key	No. Lines Displayed	Function Key	No. Lines Displayed	Function Key	No. Lines Displayed
F1	2	F7	128	F12	4096
F2	4	F8	256	F13	8192
F3	8	F9	512	F14	16384
F4	16	F10	1024	F15	32768
F5	32	F11	2048	F16	65536
F6	64				

Entering PERUSE Commands

After PERUSE displays your jobs, it issues a prompt character, an underscore ("_"), to signal that it is ready to execute your commands. You can enter commands one for each line, or you can enter several on the same line, separated by semicolons. The maximum length of a PERUSE command line is 132 characters. Each line is terminated with a carriage return; for example:

```
_J 123
_DEL
_EXIT
```

is the same as

```
_J 123; DEL; EXIT
```

Summary of PERUSE Commands

Table 6-2 contains a summary of all PERUSE commands. Refer to the GUARDIAN Operating System User's Guide for more information on how PERUSE can help you examine your spooled jobs.

Table 6-2. Summary of PERUSE Commands

Command	Function
COPIES	Alters the number of copies for the current job.
DEL	Deletes the current job.
DEV	Displays the status of a device.
EXIT	Terminates the PERUSE session.
FC	Allows you to fix and resubmit a PERUSE command.
FIND	Finds an occurrence of a string in the current job.

Table 6-2. Summary of PERUSE Commands (Continued)

Command	Function
FORM	Changes the form name of the current job.
HELP	Displays the syntax and meaning of PERUSE commands.
HOLD	Sets the hold flag for the current job.
HOLDAFTER	Sets the hold-after-printing flag for the current job.
JOB	Displays job information and sets the current job.
LIST	Lists pages from the current job to the screen or to an output device.
LOC	Changes the routing location of the current job.
NUMCOL	Sets the number of columns displayed by the LIST command.
OPEN	Specifies a new spooler supervisor.
OWNER	Changes the owner of the current job.
PAGE	Changes and displays the page position of the current job.
PRI	Changes the printing priority of the current job.
REPORT	Changes the report name of the current job.
STARTCOL	Sets the first column to be displayed by the LIST command.
STATUS	Monitors and displays status of jobs in the system.

PERUSE COMMANDS

COPIES Command

COPIES alters the number of copies for the current job. If there is no current job, then the job most recently spooled becomes the current job.

The syntax of the COPIES command is:

```
COPIES <number-of-copies>

<number-of-copies>

    is the number of copies to be made of the current job.
    The parameter is a number from 1 through 32767.  There
    is no default.
```

Considerations

- When a job is added to the spooler system, the number of copies is 1.
- For each copy you request, a separate header page also prints (if that printer has the header bit turned on).

Example

The following example illustrates how you can obtain two printed copies of a spooled job:

```
_JOB

  JOB    STATE  PAGES  COPIES  PRI HOLD  LOCATION REPORT
J 1435   READY  30     1       4         #DEFAULT ACCOUNT FRANK

_COPIES 2; JOB

  JOB    STATE  PAGES  COPIES  PRI HOLD  LOCATION REPORT
J 1435   READY  30     2       4         #DEFAULT ACCOUNT FRANK
```

DEL Command

DEL deletes the current job from the spooler system.

The syntax of the DEL command is:

```
DEL
```

Consideration

Before you can delete a job, you must make it the current job by specifying JOB <number>.

Examples

In the first example, the current job is job 52. You can use DEL to delete it from the spooler:

```
_JOB

  JOB   STATE  PAGES  COPIES  PRI HOLD  LOCATION REPORT
J 52    HOLD   2      1       4         #LPRMT3  ACCOUNT SUE
  133   READY  5      1       4         #LP      ACCOUNT SUE

_DEL
```

Another JOB command verifies that job 52 has indeed been deleted:

```
_JOB

  JOB   STATE  PAGES  COPIES  PRI HOLD  LOCATION REPORT
  133   READY  5      1       4         #LP      ACCOUNT SUE
```

However, if you put a job on hold during an earlier PERUSE, and you later reenter PERUSE and attempt to delete it, the message "NO CURRENT JOB" is displayed:

```
:PERUSE
PERUSE - T910D12 - (01OCT84)    SYSTEM  \EAST

  JOB   STATE  PAGES  COPIES  PRI HOLD  LOCATION REPORT
  456   HOLD   20     1       4          #DEFAULT ACCOUNT SUE
_DEL
NO CURRENT JOB
```

You must first make that job the current job by using the JOB command. Only then can you delete it from the spooler:

```
_HOLD OFF; J 456

  JOB   STATE  PAGES  COPIES  PRI HOLD  LOCATION REPORT
J 456   PRINT  20     1       4          #DEFAULT ACCOUNT SUE
_DEL;J
_
```

(No jobs remain in the spooler system)

DEV Command

DEV displays the status of a specified device and lists the job queue for that device.

The syntax of the DEV command is:

```
DEV $<device-name>

$<device-name>

   is the name of a device on the spooler system.
```

Considerations

- The precise form of the $<device-name> may differ from that used with TGAL. If you don't know the proper form of the $<device-name>, you can get a complete listing of devices by exiting PERUSE and typing:

  ```
  :SPOOLCOM DEV
  ```

  This helps you find the device-name form acceptable to the spooler.

- Most of the status information is self-explanatory. FORM is the form name, if any, denoting a special print device or paper associated with that printer. The jobs in the device queue are listed in the order that they will print. The wait time is the estimated total wait time (including time to print) for each job, once the currently printing job is completed. If a job of high priority is added to the queue, all the wait times change accordingly.

Example

By requesting the status of device $LP, you can better determine how long you may have to wait to get a job printed on it:

```
 DEV $LP
DEV STATE: PRINTING      FORM:

  JOB   OWNER    PAGES  WAIT        FORM
  576   008,005  22     00:05:11
 1324   001,013  40     00:13:36
  344   007,022  21     00:19:11
```

EXIT Command

EXIT ends the PERUSE session.

The syntax of the EXIT command is:

```
E[XIT]
```

Consideration

Control Y also executes an EXIT.

Example

When you exit PERUSE, you receive a command interpreter prompt:

```
_EXIT
:
```

FC Command

FC, or Fix command, allows you to modify and resubmit the last command line entered. The FC subcommands are the same as those used for the GUARDIAN command interpreter FC command, described in Section 2 in this manual.

The syntax of the FC command is:

```
FC
```

Example

If you try to use the DEL command with a "#", you receive the message "ENTRY NOT FOUND". You can fix the command by replacing "#" with "$" and reexecuting it:

```
_DEV #LP
ENTRY NOT FOUND
_FC
_DEV #LP
.    $(press RETURN)
_DEV $LP
.(press RETURN)
DEV STATE: PRINTING      FORM:

  JOB   OWNER    PAGES  WAIT       FORM
  1435  009,013  377    00:05:55
```

FIND Command

FIND locates an occurrence of a specified string and prints the line containing the string. If there is no current job, then the job most recently spooled becomes the current job.

The syntax of the FIND command is:

F[IND] [B[OTH]] [/ <find-string> /]

B[OTH]

enhances the FIND command to display both uppercase and lowercase occurrences of the <find-string>.

<find-string>

is a set of printable ASCII characters set off by two identical separators. Many people use quotes ("), apostrophes ('), or slashes(/) as separators, but any ASCII character can act as the boundary for a <find-string>.

PERUSE scans the current job, starting at the beginning of the file, for an occurrence of the specified string.

If no string is specified, then PERUSE scans the current job for the last <find-string> given, starting from where the last occurrence of the string was found.

Considerations

- NUMCOL and STARTCOL commands affect the line displayed by FIND. FIND functions if the <find-string> is located in any area, even that excluded by a NUMCOL or STARTCOL command, but it displays only that area permitted by NUMCOL and STARTCOL.
- If no line is displayed after executing a FIND command, then there are no occurrences of <find-string> in the remainder of the job.

- FIND (the default) distinguishes between uppercase and lowercase, so, for example, "BOY" does not match "boy." If you wish to search for a <find-string> in either uppercase or lowercase, use the BOTH parameter.

- You cannot use this command to locate TGAL commands embedded in your EDIT file. FIND only locates character strings that appear in the final copy.

Examples

In this example, you wish to find errors in your TAL compilation. Once you key in the beginning sequence of the error message, you can reexecute FIND without repeating the <find-string>, and it continues to show you the next occurrence of that string:

```
_FIND /**** ERROR/
**** ERROR 49 **** Undeclared Identifier
_F
**** ERROR 27 **** Illegal syntax
```

The example below shows how PERUSE can find you a phrase regardless of whether the characters are uppercase or lowercase:

```
_F B 'NONSTOP'
on NonStop system software.
```

FORM Command

FORM alters the form name of the current job. If there is no current job, then the job most recently spooled becomes the current job. The form name of a job denotes the requirement for a special print device or paper associated with that printer.

The syntax of the FORM command is:

```
FORM [ <form-name> ]

<form-name>

   is a string of up to 16 letters, digits, and spaces.
   If you omit <form-name>, PERUSE assigns a form name
   containing all blanks.
```

Considerations

- For a job to print on a given device, both the job and the device must have the same form name.
- The assignment of form names to jobs and devices is completely arbitrary, but it is intended to prevent jobs that require special print devices or paper from being printed on the wrong device. You can assign any form name to your job, but only the system operator can change the form name of a device. Use the DEV or JOB STATUS commands to check for form name restrictions on your destination device.
- When you queue a job to the spooler, it is given a form name containing all blanks.
- All form names are automatically shifted to all uppercase.

Example

In this example, you want your job to print out on a machine loaded with blank paychecks. You send the job to the spooler, specifying the location where your form paper is loaded, as the DEV command confirms:

```
   JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT
J 1221  READY  1      1458    4          #CHKWRTR  BKKPG JANET

_DEV $CHKWRTR
DEV STATE: WAITING       FORM: PAYCHECK
```

Then you specify the same form for your job. The JOB STATUS command confirms this:

```
_FORM PAYCHECK
_J S

   JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT
J 1221  READY  1      1458    4          #CHKWRTR  BKKPG JANET

   JOB  FORM              CLOSE TIME
   1221 PAYCHECK          11 DEC 84, 09:50:40

   JOB  LOCATION          DEVICE              SEQ   COPY  PAGE
   1221 #CHKWRTR.DEFAULT  $CHKWRTR            1     1     1458
```

If you sent a job to a destination whose form name did not agree with the job, your job would not print. You would receive no information with the LOC command:

```
_LOC #LP1
_
```

and you would receive an error message when you checked with the JOB #<location-name> command:

```
_J #PRTR4.DEFAULT
NO JOBS AVAILABLE
```

In order for your job to print, you must change either the job <form-name> or the job destination so that they agree.

HELP Command

HELP describes the syntax and semantics of the PERUSE commands.

The syntax of the HELP command is:

```
HELP [ / OUT <listfile> / ] [ <command-name> | ALL ]

OUT <listfile>

   is the file that receives the help message.  If OUT
   <listfile> is not specified, then the help message is
   listed at your terminal.


<command-name>

   identifies the particular command for which you want an
   explanation.  It can be any PERUSE command.


ALL

   lists a full explanation of all PERUSE commands.  If
   you specify neither <command-name> nor ALL, then an
   abbreviated description of all PERUSE commands is
   given.
```

Examples

1. The following command displays on your terminal a list of all the PERUSE commands:

```
   _HELP

COPIES  <number-of-copies>
DEL
DEV  <device-name>
EXIT
FC
```

```
FIND  [ / <find-string> / ]
FORM  <form-name>
HELP  [ / OUT <listfile> / ] [ <command-name> | ALL ]
HOLD  [ ON | OFF ]
HOLDAFTER  [ ON | OFF ]
JOB  [ <job-number> | * | STATUS | <location-name> ]
LIST [/ OUT <listfile> /] <page-range> [ , <page-range> ] ...
LOC  [ <location-name> ]
NUMCOL  <number-of-columns>
OPEN  <supervisor-name>
OWNER  { <group-#>,<user-#> | <group-name>.<user-name> }
PAGE  [ <page-number> ]
PRI  <priority>
REPORT  <report-name>
STARTCOL  <starting-column>
STATUS  [ <delay> ]
_
```

2. The following command displays on your terminal the syntax information about the PERUSE JOB command:

```
_HELP JOB

JOB  [ <job number> | * | STATUS | <location name> ]
  If a job number is given then this command causes the
  job to become current.

  If '*' is given then the job which has the latest open
  time stamp will become current.

  If 'STATUS' is given then if the current job has not
  been defined the job with the latest timestamp becomes
  the current job.  Then the status of the current job
  including destination queue information is displayed.

  If <location name> is given then the first job in the
  <location> will become current.

  If no parameter is given then the status of all jobs
  will be printed.  The job which is current will have
  a 'J' as the first character of the status line for the
  job.  The 'current' job is the one that is used for list
  commands, etc.
```

3. To print on a printer named $LP1 syntax information about the PERUSE JOB command, enter:

```
_HELP /OUT $S.#LP1/ JOB
```

HOLD Command

HOLD sets or clears the hold flag for the current job. If there is no current job, then the job most recently spooled becomes the current job.

The syntax of the HOLD command is:

```
HOLD [ ON | OFF ]

ON

   sets the hold flag for the current job.  When no
   argument is specified, ON is assumed.

OFF

   clears the hold flag for the current job.
```

Example

When you enter PERUSE in this example, job 75 is already printing:

```
_JOB 75

   JOB    STATE   PAGES   COPIES   PRI   HOLD   LOCATION   REPORT
J 75      PRINT   15      1        4             #LP        ADMIN BILL
  1001    READY   33      1        4             #DEFAULT   ADMIN BILL
```

You place a hold on it. This causes it to stop printing and removes it from the print queue.

```
_HOLD;J

   JOB    STATE   PAGES   COPIES   PRI   HOLD   LOCATION   REPORT
J 75      HOLD    15      1        4             #LP        ADMIN BILL
  1001    READY   33      1        4             #DEFAULT   ADMIN BILL
```

When you are finished, you remove the hold, and the job once again begins printing from page 1:

```
   _HOLD OFF;J

  JOB    STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT
J 75     PRINT  15     1       4          #LP       ADMIN BILL
  1001   READY  33     1       4          #DEFAULT  ADMIN BILL
```

HOLDAFTER Command

HOLDAFTER sets or clears the hold-after-printing flag for the current job. If there is no current job, then the job most recently spooled becomes the current job. When the hold-after-printing flag of a job is on and the job has completed printing, the spooler places that job in the hold state rather than deleting it from the spooler system.

HOLD puts a job in the hold state immediately; HOLDAFTER allows it to print first.

The syntax of the HOLDAFTER command is:

```
HOLDAFTER [ ON | OFF ]

ON

   sets the hold-after-printing flag for the current job.
   When no argument is given, ON is assumed.

OFF

   clears the hold-after-printing flag for the current job.
```

Example

After entering PERUSE, you can set the hold-after-flag on a job to keep it in the spooler after it has printed. The "A" under the HOLD column indicates that the hold-after-printing flag is set:

```
    JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
    722   READY  3      1       4          #HT1     INVENT PAT

_HOLDAFTER

    JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
J 722   PRINT  3      1       4     A    #HT1     INVENT PAT
```

When the job finishes printing, it enters the HOLD state and the hold-after-printing flag is still set:

```
    JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
  J 722   HOLD   3      1       4     A    #HT1     INVENT PAT
```

At a later time, you can print another copy of this job without repeating the compiling and spooling time. Remove the HOLD, and your job enters the print queue:

```
_HOLD OFF

    JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
  J 722   PRINT  3      1       4     A    #HT1     INVENT PAT
```

When it finishes printing, it again enters the HOLD state. You can do this any number of times. When you no longer have use for the job in the spooler, either delete it, or remove the hold-after-printing flag before the last time you print it.

JOB Command

JOB sets the current job or displays the status of all jobs that you own. Most of the PERUSE commands affect only the current job. The meaning of each entry in the job display is given in the beginning of this section under "How to Run PERUSE."

The syntax of the JOB command is:

```
J[OB]  [ <job-number> | * | S[TATUS] | #<location-name> ]
```

<job-number>

is the job number of a job that you own. Unless you log on with the super ID, you are not permitted to affect another user's job. Using this parameter causes the job whose number you entered to become the current job.

*

specifies the job most recently added to the spooler system. Entering "*" causes the most recently added job to become the current job.

STATUS

returns an expanded display of the status of the current job. See the description below.

#<location-name>

is the name of a spooler location in the form #<group>.<destination>. It specifies the job most recently sent to that location. For more information on spooler locations, refer to "Routing Structure" in the GUARDIAN Operating System User's Guide.

If you do not specify an argument, then the status of all jobs that you own is displayed.

Consideration

The JOB command with the STATUS subcommand produces the display shown below for the current job:

```
JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT

   JOB  FORM            CLOSE TIME

   JOB  LOCATION        DEVICE                SEQ   COPY  PAGE
```

The first part is similar to the normal JOB command display (described earlier). The second part has three entries; their meanings are as follows:

JOB	is the job number of the current job.
FORM	is the form name of the current job.
CLOSE TIME	is the date and time the collector finished collecting data from the application. A job still spooling has "OPEN" as its closing timestamp.

The third part has six entries; their meanings are as follows:

JOB	is the job number of the current job.
LOCATION	is the complete location name of the job. If the job was routed to a broadcast group, there are multiple entries.
DEVICE	is the device associated with that location.
SEQ	is the job sequence number in the device queue. if the job is printing, its SEQ is "PRINTING." The next job to print on the device has a sequence number of 1.
COPIES	is the number of copies of the job that are to be printed.
PAGE	is the number of pages in the job.

Example

The JOB command gives you information on all jobs you currently have in the spooler, one line for each job:

```
_JOB
```

The following information is displayed:

```
    JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
J 1113  READY  100    1       4           #LLPTOY  RECEIVE ANNE
  1234  READY  15     1       4           #HT10    RECEIVE ANNE
```

If you specify one job, you can obtain further information using the STATUS parameter:

```
_J 1234;J STATUS
```

The following information is displayed:

```
    JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT
J 1234  READY  15     1       4           #HT10     RECEIVE ANN

    JOB  FORM              CLOSE TIME
    1234                   08 DEC 84, 10:19:13

    JOB  LOCATION          DEVICE                SEQ   COPY  PAGE
    1234 #HT10.DEFAULT     $HT10.#HT10           2     1     15
    1234 #HT10.HT10        $HT10.#HT10           3     1     15
```

LIST Command

LIST allows you to list all or part of the current job. If there is no current job, then the job most recently spooled becomes the current job.

The syntax of the LIST command is:

```
L[IST] [ / OUT <listfile> / ]

       <page-range> [ , <page-range> ] ... [ C ] [ O ]
```

OUT <listfile>

identifies the file or device to receive the output from this command. If it is a file, it is in the form $<volume>.<subvolume>.<filename>; if it is a device, it is in the form $S.#<location>.

If <listfile> is not specified, then output will be directed to the OUT <listfile> (usually the home terminal) specified when PERUSE was entered.

<page-range>

describes the set of pages to be listed to the <listfile> and can be

```
A[LL] | <page> [ / <page> ]
```

where <page> is

```
{ F          }  [+ <number>]
{ L          }  [- <number>]
{ *          }
{ <number>   }
```

The set of pages can be identified by absolute page numbers or with a base-offset notation. Absolute page numbering begins with the first page as page 1. Base-offset notation uses a base, which can be "F" (the first page), "L" (the last page) or "*" (the current page). An offset is then added to or subtracted from the base to define the page.

→

C

causes CONTROLs, CONTROLBUFs, and SETMODEs imbedded in the job to be written to the <listfile> along with the rest of the EDIT material. Normally, this information is suppressed. Refer to the GUARDIAN Operating System Programmer's Guide.

O

causes the data to be displayed in octal representation rather than in ASCII.

Considerations

- If no current job is defined, a LIST command implicitly causes the most recently spooled job to become the current job.
- Pressing the BREAK key while PERUSE is listing a job stops the listing.
- Pressing the RETURN key or one of the function keys displays the job by lines, rather than by pages.
- NUMCOL and STARTCOL commands affect the operation of LIST. FIND functions if the <find-string> is located in any area, even that excluded by a NUMCOL or STARTCOL command, but it displays only that area permitted by NUMCOL and STARTCOL.
- Listing out to a device without including the "C" parameter causes the printer to print one line at a time. As a result, execution of the OV command in TGAL is omitted. Issuing the LIST command with the "C" parameter enables the printer to act on all the control characters issued by TGAL.
- There are similarities between the LIST OUT command of PERUSE, the TGAL OUT parameter, and the TGAL PAUSE ON parameter. You must evaluate which advantages you value for which applications:

--TGAL with the OUT or PAUSE ON parameters has a NOWAIT option; LIST does not. Therefore, using TGAL would be preferable if the file to be listed is lengthy.

--With the TGAL OUT parameter, you must know in advance where your pages begin and end in order to specify the pages you want printed:

```
:TGAL /IN <file-name>, OUT $S.#<device> / OUT 7/8
```

--The TGAL PAUSE ON parameter lets you scan your file a page at a time, so you can find out where your pages will begin and end when printed:

```
:TGAL /IN <file-name> / PAUSE ON
```

However, you can view the material only once; if you need to refer to an earlier page, you must rely on the screen-memory capacity of your terminal. And you must execute a separate TGAL instruction to print a hard copy, because PAUSE ON returns the COMINT prompt (:) to you after you view the entire file.

--To use LIST, you use TGAL to send your file to a nonexistent location:

```
:TGAL /IN <file-name>, OUT $S.#LOOK, NOWAIT/
```

From there, you can LIST, first to the screen, then out to a device, only those pages you want to print, but you cannot do it NOWAIT:

```
_LIST 23
_LIST /OUT $S.#<device> / 23 C
```

Examples

These examples do not show the pages listed since it would require too much space. However, the effect of each example is discussed in detail.

This command lists absolute pages 23 and 30 in octal display format:

```
_LIST 23, 30 O
```

This command lists pages 15 through 35, inclusive:

```
_L 15/35
```

This command lists the second page from the end of the job:

```
_LIST L-2
```

The LIST command below shows the first page, the second page, and the next-to-last page in the job. Page 5 is not listed because the current page changes during the LIST command. See also the PAGE command.

```
_P 5
_L F, *, L-1
```

You can use /OUT $s.#<device>/ to print out all or only a few pages of an already spooled file. The example below prints out pages 1 and 16 through 20 of the job. The "C" parameter preserves the original top-of-form TGAL commands such as OV.

```
_LIST /OUT $S.#LP/ 1, 16/20 C
```

LOC Command

LOC alters the location of the current job. If there is no current job, then the job most recently spooled becomes the current job.

The syntax of the LOC command is:

```
LOC [ #<location-name> ]

#<location-name>

   is the name of the new location for the current job. If
   #<location-name> is not specified, then the current job
   is given #DEFAULT (your default printer) as its new
   location.
```

Example

In this example, you specify #HOLD as the <listfile> in order to examine the job before printing:

```
_JOB

  JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT
J 777   READY  22     1       4          #HOLD     ACCOUNT ED
```

Once you finish inspection, use the LOC command to redirect the output to the specified printer for a hard copy:

```
_LOC #LP3;J

  JOB   STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT
J 777   READY  22     1       4          #LP3      ACCOUNT ED
```

NUMCOL Command

NUMCOL alters the number of columns displayed when a job is listed.

The syntax of the NUMCOL command is:

```
NUMCOL <number-of-columns>

<number-of-columns>

   is the number of columns, counting from the left margin,
   that are to be displayed or printed.
```

Considerations

- When PERUSE is first run, the setting for NUMCOL is 0, and the width of the printing device determines the number of columns to be printed. If NUMCOL is set to a value other than 0, the value specified is used as the output width. An even value is recommended. NUMCOL cannot be set to less than zero.
- NUMCOL can be used with STARTCOL to list only a few columns of a job.

Example

In the example below, you display lines of your spooled file by pressing the F2 function key, which displays four lines:

```
  (F2)
54.    000000 0  0  STRUCT buffer ;
55.    000000 0  0  BEGIN
56.    000000 0  1     INT status,
57.    000000 0  1         reserved;
```

Using the NUMCOL command, you can stop the display of all columns to the right of the eighteenth column. You can redisplay the same lines by using the PAGE command with the "*" parameter before pressing the same function key.

```
_NUMCOL 18
_P*
_(F2)
58.     000000 0  0
59.     000000 0  0
60.     000000 0  1
61.     000000 0  1
```

OPEN Command

OPEN allows you to specify a new spooler supervisor without exiting PERUSE.

The syntax of the OPEN command is:

```
OPEN <supervisor-name>


<supervisor-name>

   is the name of the new spooler supervisor that PERUSE is
   to communicate with.  The process name of the supervisor
   can be in local or remote form.

   The local form is $<process-name>, consisting of an
   alphanumeric string of 1 to 5 characters, the first of
   which must be alphabetic.

   The remote form is \<system-name>.$<process-name>, where
   \<system-name> is an alphanumeric string of 1 to 6
   characters, the first of which must be alphabetic.

   The default spooler supervisor is $SPLS.
```

Example

You can use this command to inspect a job you have spooled to a remote system without exiting PERUSE and logging on to that remote system:

```
_OPEN \FARSYS.$SPLS
```

OWNER Command

OWNER changes the owner of the current job. If there is no current job, then the job most recently spooled becomes the current job. After issuing this command, the current job becomes the property of the specified owner and can no longer be accessed by the previous owner.

The syntax of the OWNER command is:

```
OWNER {<group-name>.<user-name>       }
      {<group-number> , <user-number>}
```

<group-name>.<user-name>

is the name of the new owner, entered without spaces on either side of the period.

<group-number> , <user-number>

is the user number of the new owner.

Consideration

The default owner of a job is the user who made the initial request to the spooler.

Examples

1. An example of the OWNER command using the <group-name>.<user-name> form is:

   ```
   ADMIN.BILL
   ```

2. An example of the OWNER command using the <group-number>,<user-number> form is:

   ```
   7,10
   ```

3. You have on hold job 454 and wish to transfer its ownership to user 7,10. This you do with the OWNER command. When you next display your jobs, job 454 no longer shows, because it now belongs to user 7,10:

```
_JOB

  JOB    STATE   PAGES   COPIES   PRI   HOLD   LOCATION REPORT
  777    READY   2       1        4            #LPRMT3  ACCOUNT JUDY
J 454    HOLD    5       1        4            #LPS     ACCOUNT JUDY

_OWNER 7,10;J

  JOB    STATE   PAGES   COPIES   PRI   HOLD   LOCATION REPORT
  777    READY   2       1        4            #LPRMT3  ACCOUNT JUDY
```

PAGE Command

PAGE displays or changes the page and line-number position of the current job. If there is no current job, then the job most recently spooled becomes the current job. Page numbers begin with 1 and do not include the header page, if any. PERUSE assumes 60 lines to a page.

The syntax of the PAGE command is:

```
P[AGE] [ <number> | F | L | * ]
```

<number>

is the new page position for the current job.

F

indicates that the new page position for the current job is the first page.

L

indicates that the new page position for the current job is the last page.

*

indicates that the new page position for the current job is the first line of the current page.

When you specify no argument, the spooler displays the page and line number position of the current job.

Considerations

- The current line can be changed using a FIND command or by listing lines.

- The pages of a spooled job include not only the pages of your spooled file, but also any error pages generated by TGAL. You can use this fact to find your TGAL errors. Refer to "Example of PERUSE Operation With TGAL" in the GUARDIAN Operating System User's Guide.

Examples

In the first example, the PAGE command tells you the line and page number of where you are located in the spooled job:

```
_PAGE
  PAGE: 7       LINE: 15
```

If you use the "*" parameter, you are repositioned to the top of the current page:

```
_P *;P
  PAGE: 7       LINE: 1
```

PRI Command

PRI alters the priority of the current job. If there is no current job, then the job most recently spooled becomes the current job. The higher the priority, the sooner a job will print.

The syntax of the PRI command is:

```
PRI <priority>

<priority>

   is a number between 0 and 7, inclusive. Zero is the
   lowest priority. The default is 4.
```

Considerations

- When a job is added to the spooler, it is given a priority of 4.
- The actual queuing algorithm used by the spooler depends on the setting of the device FIFO (first-in, first-out) switch. However, regardless of the FIFO switch, higher-priority jobs are always printed before lower-priority jobs.

  --If the FIFO switch is on, jobs of the same priority are handled on a first-come, first-served basis.

  --If the FIFO switch is off, shorter jobs of the same priority are handled before longer jobs of that priority. However, a long job does not wait indefinitely for shorter jobs.
- For a more complete description of the spooler queuing algorithm, see the System Operator's Guide.

Example

In this example, your current job has the default priority of 4. You can change this to 5 with the PRI command, putting your job ahead of other jobs in the queue that have a priority of 4 or lower.

```
_J

  JOB    STATE   PAGES   COPIES PRI HOLD   LOCATION REPORT
J 52     READY   7       1      4          #LP      ACCOUNT ELAINE

_PRI 3;J

  JOB    STATE   PAGES   COPIES PRI HOLD   LOCATION REPORT
J 52     READY   7       1      3          #LP      ACCOUNT ELAINE
```

REPORT Command

REPORT alters the report name of the current job. If there is no current job, then the job most recently spooled becomes the current job. The report name is printed in the header in large banner-sized letters at the beginning of the job.

The syntax of the REPORT command is:

```
REPORT [ <report-name> ]
```

<report-name>

is the new report name for the current job. The <report-name> is composed of up to 16 letters, digits, and blanks, and must begin with a letter.

If <report-name> is omitted, then the current job is assigned a report name containing all blanks.

NOTE

The first eight characters of <report-name> are used as the first line of the banner heading characters on the header page; characters 9 to 16 appear on the second line.

Considerations

- Normally, when a job is added to the spooler, its report name is the group and user name of the owner. Refer to the Spooler Programmer's Guide for exceptions.
- Report names are automatically shifted to uppercase.

Example

After sending your job to the spooler, you want to change the report name as it will appear in the header message. Notice that you must insert spaces so that the two-name header is properly printed on two lines.

```
_JOB

  JOB   STATE PAGES COPIES PRI HOLD LOCATION REPORT
J 123   HOLD  2     1      4   #HOLD    ACCOUNT FRANK

_REPORT NEW    FORECAST
_JOB

  JOB   STATE PAGES COPIES PRI HOLD LOCATION REPORT
J 123   HOLD  2     1      4   #HOLD    NEW     FORECAST
```

STARTCOL Command

STARTCOL alters the starting column number used when a job is listed. The first column is 1.

The syntax of the STARTCOL command is:

```
STARTCOL <starting-column>

<starting-column>

   is a number greater than 0 and less than or equal to
   the rightmost column in the current job.  The default
       is 1.
```

Considerations

- When PERUSE is first run, the setting for STARTCOL is 1.
- STARTCOL can be used with NUMCOL to list only a few columns of a job.
- You can use STARTCOL to view those lines of a program that are longer than 80 characters and otherwise cannot be seen on your terminal.

Example

In this example, you use the F2 function key to display four lines of your program on the terminal:

```
_(F2)

54.     000000 0  0  STRUCT buffer ;
55.     000000 0  0  BEGIN
56.     000000 0  1     INT status,
57.     000000 0  1         reserved;
```

Then you use the PAGE * parameter to reposition yourself to the first line of the page and execute a STARTCOL command to suppress the first eighteen lines of the program. This time the F2 key displays only the instruction portion of your program:

```
_PAGE *   (reposition to first line of the page)
_STARTCOL 18
_(F2)
  STRUCT buffer ;
  BEGIN
     INT status,
         reserved;
```

STATUS Command

STATUS displays the status of all available jobs each time the state of any of the jobs changes. A "C" in front of a status line means that the state of that job has changed since the last status display.

The syntax of the STATUS command is:

```
S[TATUS] [ <delay> ]

<delay>

   is the number of seconds to wait between each status
   check. The range of values permitted for this
   argument is from 1 to 32767. When no argument is
   given, PERUSE waits 10 seconds between status checks.
```

Considerations

- Immediately before a changed display is given, the spooler sends a CTRL-G to the terminal. This causes an audible beep or ding (depending on the type of terminal) to be issued from the terminal.
- A "C" in the column to the far left indicates those jobs that have changed status since the last display.
- STATUS is a continuous display; to exit STATUS, press the BREAK key.

Example

After sending job 639 to the spooler using the NOWAIT option, you entered PERUSE and put the job on hold. When you execute the STATUS command, your cursor disappears, your terminal beeps, and a new status display appears. This will occur each time the status of a job changes.

```
_STATUS

  JOB    STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
  633    READY  7      1       4          #LP      PAYROLL GENE
J 639    OPEN          1       4          #HOLD    PAYROLL GENE

  (BEEP!!)
```

Job 639 completes spooling and now has a page count. The "C" below indicates that this job has changed status since the last screen:

```
  JOB    STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
  633    READY  7      1       4          #LP      PAYROLL GENE
JC639    READY  5      1       4          #HOLD    PAYROLL GENE

  (BEEP!!)
```

Job 633 is ahead of Job 639 in the printer queue and now begins printing. The "C" moves to indicate the most recent status change:

```
  JOB    STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
 C633    PRINT  7      1       4          #LP      PAYROLL GENE
J 639    READY  5      1       4          #HOLD    PAYROLL GENE

  (BEEP!!)
```

Job 633 finishes printing and no longer appears on the status display. Job 639 remains at location #HOLD. You can get your PERUSE prompt back by pressing the BREAK key.

```
  JOB    STATE  PAGES  COPIES  PRI  HOLD  LOCATION REPORT
J 639    READY  5      1       4          #HOLD    PAYROLL GENE

  (BREAK key)
_
```

SECTION 7

SPOOLCOM

SPOOLCOM is an interactive or noninteractive program that gives you control of the spooler. It allows all users:

- To display the status of collectors, devices, jobs, print processes, routing structure, and the spooler itself
- To change the location, state, or any attribute of your job or to delete it from the spooler system
- To restart a device that has gone offline with a device error

Information on how to use SPOOLCOM interactively and noninteractively to perform the above tasks is described in the GUARDIAN Operating System User's Guide.

Other operations performed using SPOOLCOM have great impact on the entire spooler system. For this reason, only system operators and members of the super group (with a user ID of 255,nnn) can perform these tasks, which are described in the System Operator's Guide.

This section contains the complete syntax, considerations, and examples of SPOOLCOM commands for both super-group users and others. SPOOLCOM error messages appear in the System Messages Manual.

RUNNING SPOOLCOM

SPOOLCOM resides in a file designated $SYSTEM.SYS<nn>.SPOOLCOM for Tandem NonStop systems.

You usually run SPOOLCOM from the GUARDIAN command interpreter. The syntax for the command to run SPOOLCOM is:

```
SPOOLCOM [ / <run-options> / ] [ <supervisor> [ ; ] ]

         [ <command-list> ] [ ; <command-list> ] ...
```

<run-options>

is one or more of the standard GUARDIAN operating system RUN command options, separated by commas. The RUN command options are described in Section 2 of this manual.

If you specify the OUT option, but omit the <listfile>, SPOOLCOM suppresses its output.

<supervisor>

is the name of the supervisor that SPOOLCOM communicates with. If <supervisor> is followed by a command from the <command-list>, you must insert a semicolon (;).

Use the local SPOOLCOM, in the form SPOOLCOM (do not use the remote form, \FAR.SPOOLCOM) if you wish to communicate with a remote supervisor (in the form \FAR.$SPLS).

If <supervisor> is omitted, SPOOLCOM assumes the supervisor name is $SPLS.

→

<command-list>

is a sequence of SPOOLCOM commands separated by semicolons (;). If you specify this argument, SPOOLCOM executes the commands and terminates.

To run SPOOLCOM interactively, specify no commands here.

When run interactively, SPOOLCOM sends the following startup message to the terminal:

```
SPOOLCOM - T9101B00 - (18MAR85)    SYSTEM  \EAST
```

T9101B00 is the version of SPOOLCOM.

(18MAR85) is the release date for this version of SPOOLCOM.

\EAST is the system it is running on.

The SPOOLCOM prompt is the close parenthesis character, ")".

SPOOLCOM Command Lines

Whether command lines are entered from a terminal or read from a disc file, the maximum length is 132 characters. You can enter two or more SPOOLCOM commands on the same line if you separate them with semicolons; for example:

```
)JOB 43, STATUS; EXIT
```

obtains the status of job number 43, then exits you from SPOOLCOM.

SPOOLCOM COMMANDS

A SPOOLCOM command consists of a command word possibly accompanied by a parameter; it can be followed by one or more subcommands. The command and its parameter are separated from the subcommands by commas. Subcommands are also separated from each other by commas.

For example, to specify the report name "TAL COMPILE" for job number 35, the user enters the following SPOOLCOM command:

```
)JOB 1635, HOLD, REPORT TAL COMPILE, START
```

"JOB" is the command, and the parameter "1635" indicates that job number 1635 is being referenced. In order to change the job report name, you must put the job in the hold state. Use the subcommand REPORT to specify the report name "TAL COMPILE" for the job. (The spooler converts all keyed input to uppercase.) After changing the report name, use the subcommand START to put the job back in the device queue. (In the comparable statement in PERUSE, the HOLD and START of a job are invisible to the user.)

SPOOLCOM allows you to enter commands affecting a job, a collector, or any other spooler component on a single line or separate lines. However, each command line must be complete. For example, to enter the above subcommands on three separate lines, you must repeat the command "JOB" and the parameter "1635" on each line:

```
)JOB 1635, HOLD
)JOB 1635, REPORT TAL COMPILE
)JOB 1635, START
```

These three commands have the same effect as the single command line shown above. In fact, each subcommand can be viewed as a separate command. Each subcommand is processed left to right with each being completely processed before the next subcommand is executed. The only exception to this is the DRAIN subcommand, since it can take some time for a component to completely drain. SPOOLCOM puts the component in the drain state but does not wait for the drain to complete.

A SPOOLCOM command summary describing the features available to all users is given in Table 7-1. Following that, in Table 7-2, is a SPOOLCOM command summary describing the features available only to super-group users (user ID: 255,nnn). Note the differing functions of these commands, depending on the qualification of the user.

Table 7-1. Command Summary for All Users

Command	Function
COLLECT	Obtains the status of spooler collectors.
COMMENT	Designates a comment to be ignored by SPOOLCOM.
DEV	Controls and obtains the status of devices in the spooler system. While a device is printing a job, the owner of that job can issue a SKIP and SKIPTO subcommand on that device. Anyone can START an offline device that has not been put offline by a DRAIN subcommand issued by the operator.
EXIT	Terminates a SPOOLCOM session.
FC	(Fix command) allows edit and reexecution of a command line. This is the same as the FC command in the GUARDIAN command interpreter.
HELP	Displays the syntax of SPOOLCOM commands for user reference.
JOB	Alters attributes and changes the state of your own jobs. Also obtains the status of any jobs in the spooler system.
LOC	Displays the status of the spooler routing structure.
OPEN	Specifies the particular spooler system to which the other SPOOLCOM commands refer.
PRINT	Obtains the status of the spooler print processes.
SPOOLER	Obtains the status of the spooler.

Table 7-2. Command Summary for Super-Group Users

Command	Function
COLLECT	Specifies attributes, obtains status, and changes the state of collectors.
COMMENT	Designates a comment to be ignored by SPOOLCOM.
DEV	Specifies attributes, obtains status, and changes the state of devices.
EXIT	Terminates a SPOOLCOM session.
FC	(Fix command) allows edit and reexecution of a command line. Same as FC command in the GUARDIAN command interpreter.
HELP	Displays the syntax of SPOOLCOM commands for user reference.
JOB	Specifies attributes, obtains status, and changes the state of jobs.
LOC	Defines and modifies the spooler routing structure.
OPEN	Specifies the particular spooler system to which the other SPOOLCOM commands refer.
PRINT	Specifies attributes, obtains status, and changes the state of print processes.
SPOOLER	Starts, stops, and obtains the status of the spooler.

COLLECT Command

The COLLECT command specifies attributes, obtains the status, and changes the state of the spooler collectors. Refer to the System Operator's Guide for a description of collector states and default attributes.

Any user can obtain the status of spooler collectors.

The syntax of the COLLECT command is:

```
COLLECT [ $<process-name> ] [ , <subcommand> ] ...
```

$<process-name>

is the name of the collector to which the subcommands refer. If no collector with this name exists, a new one is created.

Only the STATUS subcommand does not require that this argument be included.

<subcommand>

is one of the subcommands listed below. If <subcommand> is omitted, then the STATUS subcommand is assumed.

BACKUP <backup-cpu>

specifies the processor (-1 through 15) that is to run the collector backup. If -1 is specified, the collector does not have a backup. The default is -1; that is, the collector has no backup. Your COLLECT command must specify the primary CPU number before specifying the backup.

CPU <cpu>

specifies the processor number (0 through 15) that is to run the collector. The default is the same CPU as the supervisor.

→

DATA <data-filename>

is the name of the GUARDIAN disc file where the collector stores jobs. You must have already created the data file (see "Considerations" below). This attribute must be specified--there is no default.

DELETE

removes the specified collector from the spooler system.

DRAIN

causes the collector to stop accepting new jobs. The collector enters the dormant state when all open jobs finish spooling.

The collector must be in the dormant state in order to change its attributes.

FILE <program-filename>

specifies the file name of the program to run the collection process. Usually this is $SYSTEM.SYSTEM.CSPOOL.

NOTE

You should avoid using the name CSPOOL for any purposes other than as the <program-filename> for the collector.

PRI <process-priority>

specifies the execution priority of the collector. The range for this parameter is from 1 to 199. The default is 145.

→

START

causes a dormant collector to become active.

STATUS [/ OUT <filename> /] [DETAIL]

OUT <filename>

indicates where SPOOLCOM is to write the collector status. If not specified, the status is written to the SPOOLCOM OUT file (usually the home terminal).

DETAIL

requests a complete list of all collector attributes. See "Considerations" below for a description of the STATUS DETAIL display.

The STATUS parameter obtains the status of all collectors in the spooler system.

If a <process-name> is not specified, then the status of all collectors in the spooler system is returned.

If a <process-name> is specified, then the status of only that collector is returned.

UNIT <unit-size>

is the number of 512-word blocks requested by the collector when it needs more disc space for a job. The range is from 2 to 32767. The default is 4. See "Considerations" below.

Considerations

- If neither a <process-name> nor a <subcommand> is specified, then SPOOLCOM displays the status of all collectors.
- Any user can get the status of a collector.

- The DETAIL parameter of the STATUS subcommand provides a complete list of collector attributes.

- Only system operators can modify collector attributes.

- The collector must be in the dormant state in order to specify or change its attributes.

- Your COLLECT command must specify the CPU parameter before the BACKUP parameter. If the specified backup CPU number is the same as the primary CPU number, no backup is created.

- If a <unit-size> smaller than the minimum is specified, the collector does not use the entire data file.

- The <data-filename> should be specified only before the collector is first started. See "Creating Data Files" in the System Operator's Guide.

- The <unit-size> should be specified only before the collector is first started. Because a collector never uses more than 8192 units, one with extremely large data files must have a large unit size. You can compute the minimum unit size for any data file by using the following equations:

$$\frac{\text{size of file in words}}{512} = \text{number of blocks in file}$$

$$\frac{\text{number of blocks in file}}{8192} = \text{minimum unit size}$$

  Round all results upward. Using the above equations, a data file of up to 16 megabytes requires a minimum unit size of 2; data files up to 32 megabytes require a minimum unit size of 4; and so on.

  If a unit size smaller than the minimum is specified, the collector does not use the entire data file.

- You declare a collector by specifying its <process-name> and initialize it by specifying its other attributes. You can do this with a single COLLECT command, or you can use multiple commands.

COLLECT STATUS Display

The STATUS subcommand can be issued at any time. It produces a display with the following headings:

```
COLLECT  STATE     FLAGS  CPU  PRI  UNIT  DATA FILE     %FULL
```

The listing headers are identified as follows:

COLLECT	is the collector name.
STATE	is the state of the collector (active, dormant, drain, or error.) If the collector is in the error state, the octal number which follows the error indicates the error condition. It is either %1000 plus a GUARDIAN file-system error number, or %100000 plus a NEWPROCESS error number.
FLAGS	shows a "D" if the collector is in debug mode.
CPU	gives the numbers of the CPUs running the primary and backup processes.
PRI	is the execution priority that the collector is running.
UNIT	is the unit size of the collector.
DATA FILE	is the collector data file.
%FULL	is the percentage of the data file actually in use. Because the collector always uses some of the data space for its own bookkeeping, the file is never completely empty, although large files may show 0 percent full when empty of jobs.

Examples for All Users

1. To show the status of all collectors on the system, enter:

```
)COLLECT, STATUS
```

   The following information is displayed:

```
COLLECT STATE FLAGS CPU   PRI  UNIT DATA FILE              %FULL
$C      ACTIVE       3 ,10 146  16  $ACCTG1.SPOOLER.DATA 0
$G      ACTIVE       4 , 5 146  8   $PUBLIC.SPOOLER.DATA 5
```

2. To show the status of only one collector:

```
)COLLECT $Q, STATUS
```

   The following information is displayed:

```
COLLECT STATE FLAGS CPU   PRI  UNIT DATA FILE              %FULL
$C      ACTIVE       3 ,10 146  16  $ACCTG1.SPOOLER.DATA 0
```

3. To find more information about that one collector, specify STATUS DETAIL:

```
)COLLECT $C, STATUS DETAIL
```

   The following information is displayed:

```
COLLECTOR: $C
    STATE: ACTIVE
    LAST ERROR: NONE
    PROGRAM FILE: $SYSTEM.SYSTEM.CSPOOL
    CPU: 3
    BACKUP: 10
    PRIORITY: 146
    DATA FILE: $ACCTG1.SPOOLER.DATA
    UNIT SIZE: 16
    ALLOCATED UNITS: 1
    TOTAL UNITS: 200
    PERCENT FULL: 0
```

Examples for Super-Group Users

1. The operator specifies that CPU 3 is the primary location (and CPU 0 is the backup) from which collector $S is to run a copy of the program located in $SYSTEM.SYSTEM.CSPOOL. This collector stores jobs on the file SPLDATA:

```
)COLLECT $S, FILE CSPOOL, DATA SPLDATA, CPU 3, BACKUP 0
```

2. The operator stops the collector from accepting any new jobs, causing it to enter the dormant state when all jobs have finished spooling:

```
)COLLECT $S, DRAIN
```

COMMENT Command

The COMMENT command is used to insert explanatory material, which is ignored by SPOOLCOM, into a SPOOLCOM command file. It is often used in OBEY files to explain the purpose of the OBEY file and what the commands in it do. OBEY files are often used by system operators as an easy-to-execute and uniform procedure to start the spooler.

The syntax of the COMMENT command is:

```
COMMENT <any-text>

<any-text>

  is ignored by SPOOLCOM.  <Any-text> is terminated by a
  carriage return or a semicolon (;).
```

Consideration

At least one blank space must separate the command COMMENT from the text.

Examples

See the OBEY files to cold and warm start the spooler that are given in the System Operator's Guide.

DEV Command

The DEV command controls and obtains the status of devices in the spooler system.

System operators can use the DEV command to delete from the spooler or add to the spooler a new device that was configured at system generation, but not included in the spooler cold start. Refer to the System Operator's Guide for a description of these tasks, as well as information on device states and default attributes.

All users can obtain the status of any device, as well as perform some of the control subcommands.

The syntax of the DEV command is:

```
DEV [ $<dev-name> ] [ , <subcommand> ] ...
```

$<dev-name>

names the device to which the subcommands refer.

Only the STATUS and XREF subcommands do not require that this argument be included.

<subcommand>

is one of the subcommands listed below. If no <subcommand> is specified, STATUS is assumed.

See Table 7-3 below for a list of which subcommands are valid in which DEV states.

ALIGN

prints on the specified device a page containing the device form alignment template (a measured ruler line). This enables you to see how many characters to a line the device can print, as well as verify the paper alignment and print quality.

→

CLEAR [DEL]

stops printing the current job. If DEL is specified, the job is deleted from the spooler system; otherwise, DEV returns the job to the end of the device queue. The device remains on line so the next job in the device print queue can begin printing.

DELETE

removes the device from the spooler system. This command works only if the device is not connected to any location (that is, has been disconnected by the LOC, DEV command) and has no jobs in its queue (that is, has been drained using the DEV DRAIN command).

DRAIN

causes the device to go offline after the job currently printing is finished.

EXCLUSIVE [OFF [!]]

specifies the ownership mode of the device.

EXCLUSIVE specifies that the print process should not close the device when it is not printing a job.

EXCLUSIVE OFF (the default) specifies that the print process should close the device when there are no more jobs waiting to be printed.

EXCLUSIVE OFF! specifies that the device is to be closed between jobs.

FIFO [OFF]

specifies the algorithm by which jobs are selected for printing on the device. The default is FIFO (first in, first out) OFF. A description of the queuing algorithm appears in the System Operator's Guide.

→

FORM [<form-name>]

specifies a form name for the device, composed of up to 16 letters, numbers, and blanks. If FORM is present but <form-name> is omitted, a blank form name is assigned.

The default is a blank form name.

HEADER [ON | OFF | BATCH]

specifies whether a standard or batch header page should appear at the beginning of every job.

HEADER [ON] (the default) specifies that a header page should print at the beginning of every job.

HEADER BATCH specifies that two header pages and three trailer pages (also containing job information) print at the beginning of every job. This enables jobs printed on accordion-fold paper to have a header page appearing on top regardless of whether the job begins on an odd or even page. Printing over the page folds on the trailer pages enables jobs to be easily separated from each other.

HEADER OFF prevents any header pages from printing.

JOB <job-num>

causes a copy of the specified job to be printed on the specified device ahead of all jobs waiting.

PARM <parameter>

specifies a device parameter that is passed to the print process controlling this device whenever the supervisor communicates with the print process. The range is -32768 to +32767. The default is 0.

→

PROCESS <process-name>

specifies the print process that controls the device.

This parameter is required--there is no default.

RESTART [OFF | <interval>]

specifies the number of seconds the device waits between automatic restart attempts. If OFF is specified, device restarts are not attempted. If RESTART (the default) is on, the default interval is 120 seconds. The interval range is 10 to 32767 seconds.

RETRY <interval>

specifies the number of seconds the print process waits before retrying a failed retryable WRITE to the device. The range is from 1 to 32767. The default is 5 seconds.

SKIP [-] <num-pages>

causes the device to skip ahead or skip back <num-pages> relative to the current page. A minus sign (-) indicates skipping backward; no sign indicates skipping forward. The device starts printing at the first line on the new page. The range is from -32768 to +32767. Users who are not super-group members can execute this subcommand on their own jobs only. Otherwise, the supervisor returns the message SECURITY VIOLATION.

SKIPTO <page-num>

causes the device to skip to the page indicated. The range is from 1 to 32767 pages. Users who are not super-group members can execute this subcommand on their own jobs only. Otherwise the supervisor returns the message SECURITY VIOLATION.

→

SPEED <lines-per-minute>

specifies the speed of the device. This subcommand is used for calculating how long a job takes to print. It has no effect on the actual speed of the device. The range is from 1 to 32767.

START

causes an offline device to become online. All users can only restart devices that have been taken offline due to a device error (for example, running out of paper).

Devices taken offline by an operator can be restarted by a super-group user only.

STATUS [/ OUT <filename> /] [DETAIL]

OUT <filename>

indicates where SPOOLCOM is to write the device status. If not specified, the status is written to the SPOOLCOM OUT file (usually the home terminal).

DETAIL

requests a complete list of all of the device attributes.

The STATUS parameter displays the attributes of devices in the spooler system. See the description of the display later in this subsection.

If <dev-name> is not specified, then the status of all devices in the spooler system is returned.

If <dev-name> is specified, then the status of only that device is returned.

→

SUSPEND

causes the device to stop all activity. If the device is printing a job, the same job resumes printing when the device is restarted.

TIMEOUT <num-retries>

specifies the number of times the spooler retries a failed write to the device. The range is from 1 to 32767, or it can be -1. A -1 means the spooler retries a failed write indefinitely. The default is 360.

TRUNC [OFF]

specifies whether lines longer than the device width are to be truncated or wrapped around.

TRUNC (the default) causes the print process to discard the ends of spooled lines that are greater than the device width. TRUNC OFF causes the print process to print the ends of any spooled lines that are greater than the device width on the next sequential line.

WIDTH <device-width>

specifies the maximum line length for the device. A value of -1 (the default) causes the print process for the device to obtain the record size from the GUARDIAN file-system DEVICEINFO procedure. The range is from 1 to 32767.

XREF [/ OUT <filename> /]

produces a cross-reference list of devices, locations, and print processes, ordered by device. If device name is not specified, a complete cross reference of all devices is produced.

Table 7-3 shows the device states for which each DEV subcommand is valid. Device states are further described in the System Operator's Guide.

Table 7-3. Valid Device States During Which DEV Subcommands Can Be Used

Device States:	(O) Offline,	(B) Busy,	(W) Waiting,	(S) Suspended,	(D) Deverror,	(P) Procerror

ALIGN:	O	B	W	S	D	P	PROCESS:	O					P
CLEAR:		B		S	D		RETRY:	O					
DELETE:	O						SKIP:		B		S	D	
DRAIN:	O	B	W	S	D	P	SKIPTO:		B		S	D	
EXCLUSIVE:	O						SPEED:	O					
FIFO:	O						START:	O			S		
FORM:	O						STATUS:	O	B	W	S	D	P
HEADER:	O						SUSPEND:		B		S	D	
JOB:	O	B	W	S	D	P	TIMEOUT:	O					

Considerations

- If neither <dev-name> nor <subcommand> is specified, SPOOLCOM displays the status of all devices.
- Any user can obtain the status of a device.
- The SKIP and SKIPTO subcommands can be executed by the owner of the job currently printing on the device.
- The START subcommand can be executed by a user only if the device is offline and has a device error against it. Unless these two conditions are true, the standard user cannot use this subcommand.
- Only a system operator can set and modify the attributes of a device.
- The PARM <parameter> is passed to a print process, which can then take an application-defined action. The standard print process does not use this subcommand and parameter.

Tandem reserves the right to use the PARM <parameter> at some time in the future.

- Before deleting a device, the device queue must be empty (do a SPOOLCOM DEV, DRAIN). Either disconnect the device from its location (with the SPOOLCOM LOC, DEV command), or delete the location (with the SPOOLCOM LOC, DELETE command).

DEV STATUS Display

The STATUS subcommand produces a display with the following headings:

```
DEVICE                    STATE          FLAGS  PROC    FORM
```

The meanings of the entries are as follows:

DEVICE — is the name of the device whose status is being displayed.

STATE — is the state of the device (busy, waiting, offline, suspended, deverror, or procerror).

FLAGS — is any or all of the following flags listed here:

H	header on
F	FIFO on
E	exclusive on
D	draining
T	truncate on
!	exclusive off

PROC — is the name of the device print process.

FORM — is the device form name.

Examples for All Users

1. You can obtain the status of one device on the system by entering:

```
)DEV $HT1
```

   The following information is returned:

```
DEVICE                STATE          FLAGS  PROC    FORM
$HT1                  WAITING            T  $SPLA
```

2. Below is the detailed status of that same device:

```
)DEV $HT1, STATUS DETAIL
```

   The following information is returned:

```
DEVICE: $HT1
    STATE: WAITING
    LAST ERROR: NONE
    EXCLUSIVE: OFF
    FIFO: OFF
    HEADER: OFF
    TRUNCATION: ON
    DRAINING: NO
    PRINTING JOB: NONE
    PARM: 0
    PROCESS: $SPLA
    RETRY: 5
    TIMEOUT: 3600
    SPEED: 60
    WIDTH: 84
    FORM:
    RESTART: 120
```

3. If the job currently printing is yours, you can use the SKIPTO parameter to skip ahead in your spooled program to a page number (found by using PERUSE).

```
)DEV $LP1, SKIPTO 43
```

4. The SKIP parameter enables you to skip ahead or back a specified number of pages in your program printing. For instance, if the ribbon turned, or the paper was mangled, this command causes the printer to go back and reprint the last eight pages to get a clean copy:

```
)DEV $LP3, SKIP -8
```

5. If $HT1 has gone offline because it ran out of paper, you can restart it (after putting in new paper) with this command:

```
)DEV $HT1, START
```

Examples for Super-Group Users

1. In the example below, the operator specifies the print process that is to control device $LP3 and causes all jobs printed on that device to have a header page:

```
)DEV $LP3, PROCESS $XP, HEADER ON
```

2. The operator uses the START parameter to bring a device back online that has been taken offline for any reason:

```
)DEV $LP3, START
```

3. The operator deletes printer $LP1 from a running spooler system by first draining the device and then disconnecting it from its location:

```
)DEV $LP1, DRAIN
)LOC #LP1.LP1, DEV
)DEV $LP1, DELETE
```

EXIT Command

The EXIT command terminates an interactive SPOOLCOM session.

The syntax of the EXIT command is:

```
EXIT
```

Example

After executing SPOOLCOM commands, use EXIT to return to the GUARDIAN command interpreter:

```
)JOB 1855, HOLD, LOC #HT1, START
)EXIT
:
```

FC Command

FC, or Fix command, allows you to modify and resubmit the last command line entered. The FC subcommands are the same as those used for the GUARDIAN command interpreter FC command, described in Section 2 in this manual.

The syntax of the FC command is:

```
FC
```

Example

In the following example the FC command corrects a syntax error:

```
)DEV #LP, STATUS

ENTRY NOT FOUND

)FC
)DEV #LP, STATUS
.    $(press RETURN)
)DEV $LP, STATUS
.(press RETURN)

DEVICE                  STATE           FLAGS   PROC        FORM
$LP                     WAITING          T      $SPLP
```

HELP Command

The HELP command displays the syntax of the SPOOLCOM commands.

The syntax of the HELP command is:

```
HELP [ / OUT <filename> / ] [ <command> | ALL ]
```

<filename>

specifies the file to which the syntax is written. If omitted, the syntax description is written to the home terminal.

<command>

is any SPOOLCOM command. SPOOLCOM responds by displaying the detailed syntax of that command.

Omitting <command> causes the abbreviated syntax of all commands to be displayed.

ALL

causes the detailed syntax of all commands to be displayed.

Examples

1. To display on your terminal a list of all the SPOOLCOM commands, enter:

```
)HELP
```

   The following information is displayed:

```
COLLECT <process name>
COMMENT  [ <any text> ]
DEV <dev id>
EXIT
FC
HELP [ / OUT <filename> / ] [ <command> or ALL ]
JOB [ <job number> | ( <qualifier> [,<qualifier> ... ] ) ]
LOC  [ [<group>][.<dest>] ]
OPEN <spool name>
PRINT <process name>
SPOOLER
```

2. To display on your terminal the syntax information about the SPOOLCOM COLLECT command, enter:

```
)HELP COLLECT
```

   The following information is displayed:

```
COLLECT <process name> ,
  BACKUP <backup cpu>           !  -1:15
  CPU    <cpu>                  !  0:15
  DATA   <data filename>
  DELETE
  DRAIN
  FILE   <program filename>
  PRI    <process priority>     ! 1:199
  START
  STATUS [ / OUT <file name> / ] [ DETAIL ]
  UNIT   <unit size>            ! (1 unit = 512 words)
```

3. To send to printer $LP1 detailed syntax information about the SPOOLCOM JOB command, enter:

```
)HELP /OUT $S.#LP1/ JOB
```

JOB Command

The JOB command alters job attributes, obtains the status of jobs, and changes the status of jobs. Refer to the System Operator's Guide for a description of job states and default attributes.

All users can perform these operations on their own jobs only.

The syntax of the JOB command is:

```
JOB [ <job-num> | ( <qualifiers> ) ] [ , <subcommand> ] ...
```

<job-num>

is the number of the job referenced by the subcommands.

<qualifiers>

is any number or combination of the following qualifiers, separated by commas, and all enclosed in a parentheses. Omitted qualifiers are not considered in the selection of jobs. Only one occurrence of each qualifier is allowed.

```
OWNER [ <group-num>,<user-num>   ]
      [ <group-name>.<user-name> ]
STATE <job-state>
LOC <group>[.<destination>]
FORM <name>
REPORT <name>
COLLECT <name>
PAGES { > | < } <pages>
DATE { FROM <time> [ THRU <time>] | THRU <time>}
```

If neither <job-num> nor <qualifier> is specified, the spooler displays the status of all jobs on the system.

OWNER

defaults to the creator accessor ID of the user. OWNER can be either group and user name or number.

→

STATE

can be OPEN, READY, HOLD, or PRINT.

PAGES

allows two special cases. If PAGES > 0 is specified, all jobs are qualified. If PAGES < 0 is specified, jobs with zero pages are qualified. This makes PAGES < 0 equivalent to PAGES < 1.

DATE

If DATE is omitted, today's date is used; if <time> is omitted, all jobs for that date are qualified.

<time> is one of the following:

```
<month> <day> <year> [, <hour> : <minute> ]
<day> <month> <year> [, <hour> : <minute> ]
<hour> : <minute>
```

in the format:

```
<MMM dd yyyy> [ , <hh> : <mm> ]
<dd MMM yyyy> [ , <hh> : <mm> ]
<hh> : <mm>
```

The DATE FROM <time> THRU <time> can be used to bracket a range, such as:

```
DATE FROM DEC 16 1984, 0:0 THRU 29 DEC 1984
```

<subcommand>

is one of the subcommands listed below. If no <subcommand> is specified, then STATUS is assumed. Users who are not super-group members can use these subcommands to change the attributes and status of their own jobs only.

See Table 7-4 below for a list of the subcommands that are valid with which JOB states.

→

COPIES <num-copies>

specifies the number of copies to be printed. A job must be in the hold state to change its copies attribute. The range is 1 to 32767. The default is 1.

DELETE [!]

deletes a job from the spooler.

If you enter a JOB DELETE command on a job that is currently printing, SPOOLCOM may indicate the job has been deleted before it actually is. This is because the supervisor can delete the job only after the print process has stopped printing the job.

If a <qualifier> list is used with the DELETE subcommand, the user must verify each deletion. A response of "Y" or "y" is required to delete a job. Any other response is negative. If DELETE ! is entered, the user is not given the opportunity to verify each deletion. When deleting by job number, the "!" has no effect.

FORM [<form-name>]

specifies a form name for the job. A job must be in the hold state to change its form name. A form name is composed of up to 16 letters, numbers, and blanks.

The default is all blanks.

HOLD

places the job in the hold state. A job must be in the hold state to modify most attributes.

To remove a job from the hold state, use the START subcommand.

→

HOLDAFTER [OFF]

sets the hold-after-printing flag. HOLDAFTER causes the job to be placed in the hold state after it finishes printing. HOLDAFTER OFF (the default) allows the job to be deleted after printing.

LOC [#<location>]

specifies a new location for the job. The job must be in the hold state to change its location attribute. If the #<location> parameter is omitted, then #DEFAULT is the new location.

OWNER { <group-name>.<user-name> }
{ <group-num> , <user-num> }

designates a new owner of a job. You can use the local form of either the group and user name or the group and user number. The range for <group-num> and <user-num> is 1 to 255.

REPORT [<report-name>]

changes the report name for the job. A job must be in the hold state to change its report name. A <report-name> is composed of up to 16 letters, numbers, and blanks. It prints on the header page, 8 characters to a line.

The default report name is the user ID, that is, <group-name>.<user-name>.

SELPRI <selection-priority>

specifies the selection priority of the job. A job must be in the hold state to change its priority attribute. When a job is added to the spooler, it is given a default priority of 4 (range 0 through 7). The higher the priority, the sooner the job prints.

→

START

takes a job out of the hold state and places it in the ready state.

STATUS [/ OUT <filename> /] [DETAIL]]

OUT <filename>

indicates where SPOOLCOM is to write the job status. If not specified, the status is written to the SPOOLCOM OUT file (usually the home terminal).

DETAIL

requests a complete list of job attributes.

This parameter displays the attributes of jobs in the spooler system.

If a <job-num> is not specified, then the status of all jobs in the spooler system is returned.

If a <job-num> is specified, then the status of only that job is returned.

Table 7-4 shows the job states for which each JOB subcommand is valid. Job states are further described in the System Operator's Guide.

Table 7-4. Valid Job States During Which JOB Subcommands Can Be Used

Job States:	(H) Hold	(P) Print	(R) Ready	(O) Open
COPIES	H			
DELETE:	H	P	R	
FORM:	H			
HOLD:	H	P	R	
HOLDAFTER:	H	P	R	O
LOC:	H			
REPORT:	H			
SELPRI:	H			
START:	H			

Considerations

- System operators can change the attributes of any job.
- If neither a <job-num> nor a <subcommand> is specified, then SPOOLCOM displays the status of all jobs.
- Only users whose user ID matches the user ID of the job can change the attributes of that job, but anyone can use the STATUS subcommand.

JOB STATUS Display

The STATUS subcommand produces a display with the following headings:

```
JOB  STATE  FLAGS  OWNER  TIME  COPY  PAGE  REPORT  LOCATION
```

The explanation of the entries is as follows:

JOB — is the job number.

STATE — is the state of the job (READY, HOLD, OPEN, PRINT).

FLAGS — are any or all of the following:

- 0-7 Job selection priority
- A Holdafter on
- B Hold on (but job is still open)

X Job is bad (examples: TGAL abended while spooling the job, or the job was spooled at level 3, but the file to the collector was closed before calling SPOOLEND).

OWNER is the job owner in the form <group-num>.<user-num>.

TIME is the month and day of job creation, that is, when the job was sent to the spooler. If it was today, the time is given in <hour>:<minute> form instead.

COPY is the number of copies to be printed.

PAGE is the number of pages in the job.

REPORT is the job report name.

LOCATION is the #<group> destination for the job.

If the status of a single job is requested, the following display is also shown:

```
JOB  LOCATION        DEVICE                    SEQ   COPY  PAGE
```

The explanation of the entries is as follows:

JOB is the job number.

LOCATION is the complete destination for the job.

DEVICE is the name of the device that will print or is printing the job.

SEQ is the job sequence number in the device queue.

COPY is the same as above.

PAGE is the same as above.

If detailed status of a single job is requested, the following information is shown:

JOB is the same as above.

STATE is the same as above.

LOCATION is the #<group>.<destination> for the job.

FORM is special forms paper, if any.

REPORT	is the report name, if any.
HOLD BEFORE PRINT	is yes or no.
HOLD AFTER PRINT	is yes or no.
ABNORMAL	is yes or no.
SELECTION PRIORITY	is the same as FLAGS number above.
PAGE SIZE	is the number of lines for each page.
CREATOR ACCESS ID	is the same as OWNER above.
COPIES	is the same as above.
PAGES	is the number of pages in the report.
TOTAL LINES	is the total number of effective lines in the report.
OPEN TIME	is the time when the spooler began spooling this file.
CLOSE TIME	is the time when the spooler finished spooling this file.
DATA FILE	is the name of the collector data file.
COLLECTED BY	is the name of the collector.
UNITS ALLOCATED	is the amount of disc memory used.

Examples for All Users

1. You can enter PERUSE to find the job number of your spooled job, or you can use this SPOOLCOM equivalent:

```
)JOB (DATE FROM 10:00 THRU 11:00, OWNER)
```

The following information is displayed:

```
JOB  STATE FLAGS OWNER TIME  COPY PAGE REPORT      LOCATION
1636 READY 4     4,19  10:18 1    1    PUBLCTY PAT #UNTIL
```

2. Then you can use the JOB command with the job number to give you status information:

```
)JOB 1636
```

The following information is displayed:

```
JOB  STATE FLAGS OWNER TIME  COPY PAGE REPORT       LOCATION
1636 READY 4     4,19  10:18 1    1    PUBLCTY PAT #UNTIL

 JOB  LOCATION          DEVICE              SEQ   COPY  PAGE
 1636 #UNTIL.DEFAULT                         4     1     5
```

3. The DETAIL parameter gives even more job information:

```
)JOB 1636, STATUS DETAIL
```

The following information is displayed:

```
JOB: 1636
    STATE: READY
    LOCATION: #UNTIL
    FORM:
    REPORT: PUBLCTY PAT
    HOLD BEFORE PRINT: NO
    HOLD AFTER PRINT: NO
 ABNORMAL: NO
 SELECTION PRIORITY: 4
 PAGE SIZE: 60
 CREATOR ACCESS ID: 4,19
 COPIES: 1
 PAGES: 5
 TOTAL LINES: 60
 OPEN TIME: 27 MAR 84, 10:18:48
 CLOSE TIME: 27 MAR 84, 10:18:51
 DATA FILE: $SPOOL.SPOOLER.DATA
 COLLECTED BY: $S
 UNITS ALLOCATED: 1
```

4. In the following example, the JOB command is used to alter job attributes. The job is placed in HOLD to make the changes, the hold-after-printing flag ("A") is set; the report name for the header page is altered; and a status show is again requested, followed by the START subcommand to remove the job from HOLD:

```
)JOB 36, HOLD, HOLDAFTER, REPORT FINANCE REPORT, STATUS, START
```

The following information is displayed:

```
JOB  STATE FLAGS OWNER TIME  COPY PAGE REPORT         LOCATION
  36 HOLD  4 A   4,19  10:18 1    5    FINANCE REPORT #UNTIL
```

3. Placing the job on hold truncates the display, as shown above. You must specify the STATUS parameter after removing the hold in order to see the two-line display:

```
JOB  STATE FLAGS OWNER TIME  COPY PAGE REPORT         LOCATION
  36 READY 4 A   4,19  10:18 1    1    FINANCE REPORT #UNTIL

  JOB  LOCATION          DEVICE                SEQ   COPY  PAGE
   36  #UNTIL.DEFAULT                           1     1     1
```

Examples for Super-Group Users

Super-group users can perform any of the functions described above on any spooled jobs, not just their own.

LOC Command

The LOC command defines and modifies the spooler routing structure. Refer to the System Operator's Guide for a description of default location attributes and a description of the spooler routing structure.

All users can display the status of the spooler routing structure (with the STATUS subcommand) or produce a cross-reference list of locations, devices, and print processes (with the XREF subcommand).

In addition, members of the super group can specify BROADCAST modes, delete a location (DELETE), and connect or disconnect a device from a location (DEV).

Briefly, the syntax of the LOC command is:

```
LOC [ <location> ] ,    STATUS
                        XREF
                        BROADCAST
                        DELETE
                        DEV


<location>

   is the logical destination of a job.  If a print device
   is associated with a specific <location>, that print
   device becomes the physical destination of the job.
   <location> is a two-part name:  #<group>.<dest>, as
   described under "Routing Structure" in the System
   Operator's Guide.  If either name part is omitted, it
   takes its default value.
```

The detailed syntax of the LOC command depends on the particular subcommand, described as follows, in the same subcommand order.

The syntax of the LOC, STATUS subcommand is:

```
LOC [ #<group> | <dest> | #<group>.<dest> ] ,

    [ STATUS ] [ / OUT <filename> / ] [ DETAIL ]
```

STATUS

displays the attributes of locations in the spooler system. If the XREF, BROADCAST, DELETE, or DEV subcommands are not specified, the default is STATUS.

OUT <filename>

indicates where SPOOLCOM is to write the location status. If not specified, the status is written to the SPOOLCOM OUT file (usually the home terminal).

DETAIL

requests a complete list of all of the attributes for all locations. You must specify STATUS if you use the DETAIL parameter.

If #<group> and <dest> are specified, the status of only that particular location is returned.

If <dest> is not specified, the status of all destinations associated with the group is returned.

If #<group> is not specified, the status of all locations in the routing structure is returned.

Considerations. The following points apply to the LOC, STATUS command.

- If neither a <location-name> nor a <subcommand> is specified, then SPOOLCOM displays the status of all locations.
- Any user can obtain the status of any location in the spooler system.

- Using the STATUS subcommand without specifying #<group> or <dest> produces an alphabetic display with the following headings for all spooler locations:

```
LOCATION            FLAGS   DEVICE
```

The meanings of the entries are as follows:

LOCATION	is the #<group>.<dest> of the location whose status is being displayed.
FLAGS	displays a B if broadcast is on.
DEVICE	is the device associated with that location, if any.

- Using the STATUS subcommand with a #<group> or <dest> produces a two-line display with the following headings for each job at that location:

```
LOCATION            FLAGS   DEVICE

  JOB  LOCATION          DEVICE              SEQ   COPY  PAGE
```

The meanings of the entries are as follows:

LOCATION	is the same as above.
FLAGS	is the same as above.
DEVICE	is the same as above.
JOB	is the job numbers of all jobs at that destination.
LOCATION	is the same as above.
DEVICE	is the same as above.
SEQ	is the job position in that destination queue.
COPY	is the number of copies of that job.
PAGE	is the number of pages in that job.

- Specifying LOC, DETAIL or LOC, STATUS DETAIL produces another display:

```
LOCATION:
    BROADCAST:
    DEVICE:
```

The meanings of the entries are as follows:

LOCATION	is the same as above.
BROADCAST	is yes (ON) or no (OFF).
DEVICE	is the same as above.

Examples for All Users. Placing your job on hold removes it from any location queue; therefore, the same LOC #<group> that gave you location information before now produces no display:

```
)LOC #NEW
```

The following information is displayed:

```
LOCATION            FLAGS  DEVICE
#NEW.DEFAULT

  JOB  LOCATION           DEVICE                SEQ   COPY  PAGE
  1636 #NEW.DEFAULT                             6     1     4

)JOB 1636, HOLD
)LOC #NEW
)
```

Requesting LOC STATUS DETAIL tells you whether or not this location is part of a broadcast group:

```
)LOC #NEW, STATUS DETAIL
```

The following information is displayed:

```
LOCATION: #NEW.DEFAULT
    BROADCAST: OFF
    DEVICE:
```

The syntax of the LOC, XREF subcommand is:

```
LOC [ #<group>.[<dest>] ] , XREF [ / OUT <filename> / ]

XREF

   produces a cross-reference list of locations, devices,
   and print processes ordered by location.  If a location
   is not specified, a complete cross-reference is produced.
   If a location is specified, it can be entered as:

      #<group> | #<group>.<dest>

   If #<group> is specified, all locations within that group
   are listed.  If #<group>.<dest> is specified, a cross
   reference for that location is produced.
```

Examples for All Users. If you send your job to a real location, as in the first example below, the XREF option reveals the device name and print process name associated with it. In contrast, if the location does not exist (as in the second example), there is no device or print process associated with that fictitious location:

```
)LOC #LPR1, XREF
```

The following information is displayed:

```
LOCATION              DEVICE                          PRINT PROCESS
#LPR1.DEFAULT         $LP1                            $SPLA
```

```
)LOC #LOOK, XREF
```

The following information is displayed:

```
LOCATION              DEVICE                          PRINT PROCESS
#LOOK.DEFAULT
```

The syntax of the LOC, BROADCAST subcommand is:

LOC #<group> , BROADCAST [OFF]

specifies the broadcast mode of the group.

BROADCAST

causes a job routed to #<group> to be printed on ALL devices connected to #<group>.

BROADCAST OFF

(the default) causes jobs routed to the group to be printed on that device (connected to the group) that can print the job the fastest.

Consideration. Only system operators can modify the attributes of a location.

Example for Super-Group Users. The operator uses this command to turn broadcasting on for all devices in the group #BRODC. This causes any job routed to #BRODC to print on all the devices connected to this group:

```
)LOC #BRODC, BROADCAST ON
```

The syntax of the LOC, DELETE subcommand is:

LOC #<group>[.<dest>] , DELETE

deletes from the spooler the entire group or the particular destination within the group. This works only if there are no jobs currently in the location being deleted.

Consideration. Only system operators can modify the attributes of a location.

Example for Super-Group Users. To delete printer $LP8 from the spooler, enter the command:

```
)LOC #LP8, DELETE
```

The syntax of the LOC, DEV subcommand is:

LOC [#<group>].<dest> , DEV [<device-name>]

connects or disconnects a location to a device.

If #<group>.<dest> is specified, the command refers to that particular group and destination. If only the <dest> is present, then the command refers to every group that has that destination in it.

If <device-name> is present, the command establishes a connection; if <device-name> is absent, any existing connection with a device is broken.

Consideration. Only system operators can modify the attributes of a location.

Example for Super-Group Users. With this command, the operator connects device $LP to the location #PRIN.DEFAULT:

```
)LOC #PRIN.DEFAULT, DEV $LP
```

OPEN Command

The OPEN command specifies the spool supervisor with which SPOOLCOM communicates.

The syntax of the OPEN command is:

```
OPEN <spool>

<spool>

   is the process name of a supervisor in remote or local
   form.
```

Considerations

- When running SPOOLCOM, you can specify which supervisor to open. If no supervisor name is specified, then SPOOLCOM opens a file to a process named $SPLS.
- Refer to the GUARDIAN Operating System User's Guide for information on the correct syntax for a process name.

Example

You can use this command to communicate with a job you have spooled to a remote location, for example, supervisor $SPFAR on network \FARSYS:

```
)OPEN \FARSYS.$SPFAR
```

PRINT Command

The PRINT command specifies attributes, obtains the status, and changes the status of the spooler print processes. Refer to the System Operator's Guide for a description of print-process states and default attributes.

All users can obtain the status of any spooler print processes.

The syntax of the PRINT command is:

```
PRINT [ $<process-name> ] [ , <subcommand> ] ...
```

$<process-name>

is the process name of the print process whose attributes are to be specified by the PRINT subcommands.

Only the STATUS subcommand does not require that this argument be included.

<subcommand>

is one of the subcommands listed below. If no <subcommand> is specified, then STATUS is assumed.

BACKUP <backup-cpu>

is the processor (-1 through 15) that runs the print process backup. The default is -1, which specifies that the print process has no backup. At this time, the print process supplied by Tandem does not use the BACKUP subcommand. See "Considerations."

If a <backup-cpu> is specified and a print process primary CPU goes down, the spool supervisor restarts the print process, as well as any of its devices for which DEV RESTART is on. A stopped process is not restarted. At restart time, the CPU and the backup CPU are switched. Operation of user-written print processes is not affected by this switch.

→

CPU <cpu>

is the processor number (0 through 15) that runs the print process. The default is the same CPU as the supervisor.

DEBUG [OFF]

sets the debug mode of the print process.

DEBUG specifies that this print process runs in debug mode; DEBUG OFF (the default) specifies that the print process does not run in debug mode. See "Debugging Print Processes" in the Spooler Programmer's Guide. Also see "Considerations."

DELETE

removes the print process from the spooler system. You must place the print process in the dormant state before you can delete it.

FILE <program-filename>

specifies the program file for this print process.

If omitted, the supervisor assumes that this is an independent print process. See "Considerations."

PARM <parameter>

is a print-process parameter that is passed by the supervisor to the print process in the startup message. Its meaning is defined by a user-written print process. The standard print process does not use this parameter. The range is -32768 to +32767. The default is 0.

→

PRI <execution-priority>

specifies the execution priority of the print process. The range for this parameter is 1 to 199. The default is 145.

START

takes a print process out of the procerror state after the cause of the failure has been remedied.

STATUS [/ OUT <filename> /] [DETAIL]]

OUT <filename>

indicates where SPOOLCOM is to write the print-process status. If not specified, the status is written to the SPOOLCOM OUT file (usually the home terminal).

DETAIL

requests a complete list of all of print-process attributes.

The STATUS parameter displays the attributes of print processes in the spooler system.

If a print process is not specified, then the status of all print processes in the spooler system is returned. If a print process is specified, then the status of only that print process is returned.

XREF [/ OUT <listing-device> /]

XREF produces a cross-reference listing of print processes, devices, and locations, ordered by print process. If a print process is not specified, a complete cross-reference is produced. (This takes the most time of any XREF commands.)

Considerations

- If neither a <process> name nor a <subcommand> is specified, then SPOOLCOM displays the status of all print processes.
- Any user can obtain the STATUS of print processes in the spooler system.
- Only members of the super group can initialize and set the attributes of a print process.
- An independent print process is one that is running when the spooler is started. The spooler does not start it and assumes that it is always running. Refer to the Spooler Programmer's Guide for more information on independent print processes.
- Attributes of a print process can be specified only when the print process is in the dormant or procerror state.
- The supervisor runs print processes only as needed. A print process controlling an exclusive device runs all the time, while a print process controlling only shared devices runs when the print process is actually printing a job on one of its devices.
- The print process supplied by Tandem does not run as a NonStop process, so the BACKUP subcommand is ignored. Tandem reserves the right to use this subcommand in the future.
- When a print process is in debug mode, it is not timed out by the spooler. This means that the spooler waits indefinitely for a response. For this reason, print processes should never be debugged on a production spooler.

PRINT STATUS Display

The STATUS subcommand produces a display with the following headings:

```
PRINT    STATE          FLAGS  CPU    PRI
```

The explanation of the entries is as follows:

PRINT — is the name of the print process whose status is being given.

STATE — is the state of the print process (dormant, active, or procerror).

FLAGS can be either or both of the following entries:

I Process is an independent process.
D Process is in debug mode.

CPU gives the numbers of those CPUs running the primary and backup processes.

PRI is the execution priority at which the print process runs.

When DETAIL is specified, the following information is shown:

PRINT PROCESS is the same as above.

STATE is the same as above.

LAST ERROR is the octal number of the last error printed on the error-log file. It is either %1000 plus a GUARDIAN file-system error number or %100000 plus a NEWPROCESS error number.

DEBUG is the same as FLAG D above.

INDEPENDENT is the same as FLAG I above.

PROGRAM FILE is the location of the file where the print process resides.

CPU and BACKUP is the same as CPU above.

PRIORITY is the same as PRI above.

PARM is a print-process parameter.

Examples for All Users

1. To display the status of a print process, enter:

```
)PRINT $SPLA
```

The following information is displayed:

```
PRINT    STATE          FLAGS  CPU    PRI
$SPLA    ACTIVE                5 , 9  145
```

2. To display the status of that same print process in greater detail, enter:

```
)PRINT $SPLA , STATUS DETAIL
```

The following information is displayed:

```
PRINT PROCESS: $SPLA
    STATE: ACTIVE
    LAST ERROR: NONE
    DEBUG: OFF
    INDEPENDENT: NO
    PROGRAM FILE: $SYSTEM.SYSTEM.PSPOOL
    CPU: 5
    BACKUP: 9
    PRIORITY: 145
    PARM: 0
```

Examples for Super-Group Users

1. The operator can use the PRINT command to define the print process $XP to be a copy of the program file $SYSTEM.SYSTEM.PSPOOL and to run on CPU 3:

```
)PRINT $XP, FILE PSPOOL, CPU 3
```

2. Then the operator can start the print process $XP, moving it out of the procerror state, ready to print jobs.

```
)PRINT $XP, START
```

SPOOLER Command

The SPOOLER command starts, stops, and obtains the status of the spooler. Refer to the System Operator's Guide for a description of spooler states.

All users can obtain the status of the spooler with this command.

The syntax of the SPOOLER command is:

```
SPOOLER [ , <subcommand> ] ...
```

<subcommand>

is one of the subcommands listed below. If no <subcommand> is specified, then STATUS is assumed.

DRAIN

brings the spooler to an orderly halt after all jobs that are currently printing or spooling have completed. This subcommand moves the spooler from the active to the dormant state.

ERRLOG <filename>

specifies a new log file for the spooler error messages.

If <filename> is omitted, logging is turned off.

START

starts the spooler (collectors and print processes). You can start the spooler when it is in the warm or the cold state.

→

```
STATUS [ / OUT <filename> / ] [ DETAIL ]

   OUT <filename>

      indicates where SPOOLCOM is to write the status
      information.  If not specified, the information is
      written to the SPOOLCOM OUT file (usually the home
      terminal).

   DETAIL

      requests a complete list of all status information.

   The STATUS parameter displays the status of the spooler
   system.  See "Considerations."
```

Considerations

- Any user can obtain the status of the spooler.
- Only members of the super group can modify the attributes of the spooler.
- The DRAIN command stops the spooler in an orderly manner. It is the only recommended way to stop the spooler. Following the SPOOLER DRAIN command:

  --The collectors allow current jobs to finish but reject new opens with a GUARDIAN file-system error 66 (DEVICE DOWNED BY OPERATOR). Each collector stops when it has no more open jobs.

  --Each print process finishes printing any active jobs, then stops.

  --After all collectors and print processes have stopped, the supervisor stops.

  --The spooler enters the dormant state, ready to be warm started.

SPOOLER STATUS Display

The STATUS subcommand produces a display with the following headings:

```
SPOOLER  STATE      LOGGING FILE                LAST ERROR
```

The explanation of the entries is as follows:

SPOOLER	is the process name of the spooler.
STATE	is the state of the spooler (active, drain, warm, or cold). SPOOLCOM cannot communicate with a dormant spooler.
LOGGING FILE	is the spooler error-log file.
LAST ERROR	is the last error sent to the log file.

Example for All Users

Using the SPOOLER command, you can obtain the status of your spooler:

```
)SPOOLER
```

The following information is displayed:

```
SPOOLER          STATE      LOGGING FILE          LAST ERROR
$SPLS            ACTIVE     $0
```

Example for Super-Group Users

The operator uses the SPOOLER command to start the spooler, including any collectors and print processes associated with it:

```
)SPOOLER, START
```

SECTION 8

DISC SPACE ANALYSIS AND DISC COMPRESSION UTILITIES (DSAP AND DCOM)

The Disc Space Analysis Program (DSAP) analyzes how disc space is being utilized on a specified volume, and the Disc Compression Program (DCOM) moves files in order to gain more usable space on the disc. Both utilities operate online while the disc is up and operating.

DSAP copies the disc directory and the disc free-space table of a given disc volume to its own working storage and then, depending on which options you specify, manipulates this data to produce from one to seven reports relevant to the use of disc space on that volume.

Refer to the System Operator's Guide for a description of how DSAP and DCOM work, as well as for background information on file-description terminology. DSAP and DCOM error messages can be found in the System Messages Manual.

SECURITY

Both DSAP and DCOM are privileged, and it is up to each customer to determine which users are allowed to run them. Accessibility can be controlled in two ways:

- Do not license the file for general use. This means only the super ID (255,255) can run the programs.
- License the file to some subset of users, setting up file security on some basis of controlled use.

When determining security, be aware that DSAP only measures. It has no effect on the system, so restricting its use is not needed. On the other hand, DCOM does have an effect on system performance--it improves it. However, while DCOM is working, it can negatively impact system performance--not greatly, but enough so that its use should be more restricted, generally to operators (that is, super-group users).

DSAP--THE SPACE ANALYZER

DSAP is a utility program designed for disc-space analysis.

DSAP is a noninteractive program. The default output file is the home terminal, though you can specify other types of OUT file, such as a disc file. When the OUT file is a terminal, the output width is 79, and the output is not formatted into pages. In other cases, such as when the output is to the spooler, an output width of 132 and a page length of 60 are assumed. A header message prints at the top of each page in the following format:

```
PAGE <n> DSAP -- $<volume> ON \<system> -- <report-name> -- hh:mm:ss
```

Refer to the System Operator's Guide for a description of how DSAP works. DSAP error messages can be found in the System Messages Manual.

DSAP Program

The syntax of the command to run DSAP is:

```
DSAP  [ / <run-options> / ] [ $<volume> | HELP ]

      [ USER [ <group-name>.<user-name>  ]                    ]
             [ <group-num>,<user-num>    ]
             [ -1                        ]
             [ <group-name>.*            ]
             [ <group-num>.*             ]
        EXTENTCHECK
        WORKFILE { $<volume> | <filename> }
        SUMMARY
        FREESPACE
        FILESPACE
        SPACE
        FILESIZE                                              ...
        BYSUBVOL
        BYUSER
        ANALYSIS
        DETAIL
        SEPARATE
        AGE [ OVER ] <n>
        AGE UNDER <n>
        SIZE [ OVER ] <n>
        SIZE UNDER <n>
        UNUSED <n>
        DEALLOC <n>
        AUDITED
        LICENSED
        PARTITIONED
        TEMPORARY
        CRASHOPENED
        PROGID
        BROKEN
        ROLLFORWARD [ NEEDED ]                                ]
```

<run-options>

are options to the GUARDIAN command interpreter RUN command, as described in Section 2. In particular, if you wish to send output to a file other than the home terminal, specify the OUT option.

→

$<volume>

specifies the disc volume on which the analysis is to be performed. If you omit $<volume>, a brief description of the DSAP program and its options is displayed (identical to that displayed if you specify HELP).

HELP

displays a brief description of the DSAP program and its options.

The option keywords and parameters listed below can follow the $<volume> specification in any order. Using commas to separate options is optional. If no options are specified, DSAP produces the summary report.

USER

limits the files analyzed to those owned by a single user or by a single group of users. USER without any qualification specifies the current user. If you omit USER, DSAP analyzes all files on the disc.

<group-name>.<user-name> | <group-num>,<user-num>

USER qualified with a group and user name or number specifies a particular user in a particular group.

-1

USER -1 specifies the super ID (255,255)

<group-name>.* | <group-num>.*

USER qualified by a group name or number and asterisk specifies all users belonging to the specified group name or group number.

→

EXTENTCHECK

requests that DSAP make a repeated effort to perform an extent consistency check. Whenever all files are being analyzed (that is, when you do not specify a USER option), DSAP ordinarily checks the consistency of the allocated and free disc extents and displays this information in the space allocation consistency analysis portion of the summary report. In order to perform this check, DSAP must be able to copy the disc directory without intervening changes. If changes occur during the copy attempt, DSAP continues the summary report without checking extent consistency, issuing the following messages:

```
Free space changed during directory copy.
Space allocations are too frequent.
Extent check cannot be performed.
```

If you do specify EXTENTCHECK, DSAP retries the directory copy three times before giving up, issuing the following messages:

```
Free space changed during directory copy.
Retrying.
```

If the directory cannot be copied without changes on any of these attempts, DSAP terminates with the following messages:

```
Free space changed during directory copy.
Space allocations are too frequent.
Extent check cannot be performed.
DSAP is being terminated.
```

DSAP does not complete the summary report.

If the disc you are checking is exceptionally busy (for example, $SYSTEM), you may want to perform EXTENTCHECK at a time when the disc is less active.

→

WORKFILE { $<volume> | <filename> }

$<volume>

specifies a volume for the allocation of a temporary file, and is of the form $<volume>.

<filename>

specifies a file name for the allocation of a permanent work file (used by ENFORM to generate additional reports) and is of the form $<volume>.<subvolume>.<filename>. See "DSAP Considerations."

If you do not specify a WORKFILE option, the DSAP work file is allocated, by default, on the volume where the program swap file resides. (Using BINDER, it is possible to alter this default permanently. See "Altering the Default Work File Volume" at the end of this section.)

Report Options: If you do not select any of the following report options, then DSAP generates only the summary report, described below.

SUMMARY

generates the summary report. If you select any reports, DSAP generates the summary report as well, whether you specifically request it or not.

If you want to see only the summary report, you can either specify SUMMARY or omit any report options. In other words,

```
:DSAP $<volume> SUMMARY
```

is the same as

```
:DSAP $<volume>
```

→

FREESPACE

generates the report on free-space distribution--an analysis of the distribution of free-space extent sizes.

FILESPACE

generates the report on file extent-size distribution--an analysis of the distribution of allocated extent sizes.

SPACE

generates both the free-space distribution and file extent-size distribution reports, that is, FREESPACE plus FILESPACE.

FILESIZE

generates the report on file-size distribution--an analysis of the distribution of file sizes.

BYSUBVOL

generates the subvolume summary report--an analysis of space allocation for each subvolume.

BYUSER

generates the user summary report--an analysis of space allocation for each user ID.

ANALYSIS

generates all analysis reports except the detail report, that is, SUMMARY, FREESPACE, FILESPACE, FILESIZE, BYSUBVOL, and BYUSER. See Table 8-1 below which relates each report name to its contents and the DSAP option that calls it.

→

DETAIL

generates the user detail report as well as the summary report.

This report lists the names of all files (in the file set under analysis) that meet at least one criterion specified by the DETAIL options selected. The list is ordered by user ID and subordered by file name.

If you specify no DETAIL options, DSAP lists all files in the file set.

If you specify one or more of the DETAIL options, you need not explicitly specify DETAIL.

SEPARATE

If you also specify the SEPARATE option, DSAP formats the user detail report with the detail report for each user beginning on a new page (so that the relevant portions can easily be distributed to each disc user).

The DETAIL options are:

AGE [OVER] <n>

causes the DETAIL report to include all files in the file set whose most recent modification occurred <n> days or more ago.

AGE UNDER <n>

causes the DETAIL report to include all files in the file set modified in the last <n> days.

SIZE [OVER] <n>

causes the DETAIL report to include all files in the file set with <n> pages or more.

→

SIZE UNDER <n>

causes the DETAIL report to include all files in the file set with fewer than <n> pages.

UNUSED <n>

causes the DETAIL report to include all files in the file set with <n> or more pages of unused space.

DEALLOC <n>

causes the DETAIL report to include all files in the file set with <n> or more pages in deallocatable extents.

AUDITED

causes the DETAIL report to include all files audited by the Transaction Monitoring Facility (TMF) in the file set.

LICENSED

causes the DETAIL report to include all licensed files in the file set.

PARTITIONED

causes the DETAIL report to include all partitioned files in the file set.

TEMPORARY

causes the DETAIL report to include all temporary files in the file set.

CRASHOPENED

causes the DETAIL report to include all files in the file set that are in the crash-opened state. A file is listed in the crash-opened state if it was open at the time of a total system crash or when the disc on which it is located became unavailable. The CRASHOPENED state is cleared whenever the file is successfully opened.

PROGID

causes the DETAIL report to include all files in the file set that have the PROGID option set. PROGID is set by the FUP SECURE command. (For further details about the FUP SECURE command, refer to Section 2.

BROKEN

selects files that are marked as broken due to a detected inconsistency in the file or a read-write I/O error. This option is applicable to DP2 volumes only; it is ignored for DP1 volumes.

ROLLFORWARD [NEEDED]

selects files audited by TMF that are marked as needing a rollfoward because a crash-recovery operation failed. This option is applicable to both DP1 and DP2 volumes.

DSAP Considerations

- DSAP has no internal security restrictions other than the fact that it requires licensing. The system manager controls access to DSAP using the standard disc file security scheme.

- In order to ensure compatibility with low-level system interfaces, DSAP requires that its version match the version of the GUARDIAN operating system.

- DSAP allocates a substantial extended data segment and swap file to hold file labels and extent information. If a specific disc is close to being full and the number of extents on that disc is large (the volume was configured for a large number of files), DCOM may attempt to map the extended data segment to a temporary file on that disc, but terminate with a GUARDIAN file-system errors 31 or 43 (UNABLE TO OBTAIN DISC SPACE). You can, therefore, specify a disc that is not as full as the work-file volume, in order to avoid this problem.

- With the WORKFILE option, you can specify either a temporary or a permanent file for this purpose. Otherwise, the program uses its swap-file volume as a default location. If, on a given system, a particular disc is preferred for large temporary files, the system manager can specify that volume as the default by modifying DSAP with BINDER. Simple instructions for making this alteration are provided below under "Altering the Default Work-File Volume".
  It is best, in any event, to specify a nonexisting work file, because the file created is then automatically the appropriate size. If an existing work file is specified, it is used, but DSAP terminates if that file is not large enough.

- You need to specify a permanent work file only if you want to retain file label information in machine-readable form. Using ENFORM or a custom program, you can then generate additional reports with more complex selection conditions than those offered by the DSAP options. For a description of ENFORM, refer to the ENFORM User's Guide.

- The order of the records in this permanent file depends on the order in which which DSAP reports your request. Initially, the records are ordered in the standard collating sequence by their names. The BYUSER and DETAIL reports sort the records by owner ID and then by file name. The format of these records in the permanent file is given in a Data Definition Language (DDL) description released with the program (in a file called DSAPDDL). For a description of DSAPDDL, refer to "Format of the Permanent Work File" below.

- You can exit a running DSAP process at any time by pressing the BREAK key. DSAP continues to run in the background until it is finished. If DSAP is sending output to your terminal, you can resume running DSAP by typing "PAUSE".

- To stop a running DSAP process, press the BREAK key and, when the COMINT prompt (:) appears, issue a STOP command. Terminating the DSAP program in the midst of compression simply leaves the disc in a partially compressed state, which is still an improvement over a totally uncompressed disc. Early termination of DSAP has no detrimental effects.

DSAP REPORT FORMATS

The formats of the seven report types are summarized in Table 8-1 below with their DSAP option name and contents description.

Table 8-1. Summary of DSAP Report Types

DSAP Reports	Report Name	Report Contents
Report Options:		
SUMMARY	Summary Report	Physical disc description Space use summary Exceptional condition report
SUMMARY USER	truncated Summary Report	Space use summary for user
FREESPACE	Free Space Distribution	Space distribution analysis of free space extents
FILESPACE	File Extent Size Distribution	Space distribution analysis of allocated file extents
SPACE	= FREESPACE + FILESPACE	Space distribution analysis of free space extents and allocated file extents
FILESIZE	File Size Distribution Report	Space distribution analysis of file sizes
BYSUBVOL	Subvol Summary Report	Space allocation for each subvol
BYUSER	User Summary Report	Space allocation for each user ID
ANALYSIS	SUMMARY FREESPACE FILESPACE FILESIZE BYSUBVOL BYUSER	} all the analysis reports as described above
DETAIL	User Detail Report	File names and types, according to DETAIL options selected
Other Options:		
USER	truncated Summary Report FILESIZE FILESIZE BYSUBVOL Temporary Files	} the reports as described above for the specified users
EXTENTCHECK	SUMMARY	Tries 3 times to add consistency check to Summary Report
WORKFILE	SUMMARY	Specifies the location for the workfile

DSAP Summary Report

The summary report for a disc volume is divided into three sections: a physical disc description, a summary of space use, and an exceptional condition report:

- The physical disc description includes device type, subtype, the approximate formatted disc capacity, and the actual usable disc capacity (the disc space available for disc directory, free-space table, volume label, disc files, and so on). The difference between the formatted and usable disc capacity is areas reserved for defect handling and maintenance.

- The summary of space use lists, for the specified disc, the total number of:

  --Free pages (available for file allocation)

  --Allocated pages (assigned to a file)

  --Unused pages (assigned to a file but beyond the current end of file)

  --Deallocatable pages (unused pages in assigned but empty extents)

  You can reclaim deallocatable space by using the File Utility Program (FUP) DEALLOCATE command to return unused extents. Refer to the Section 2 for more information about the FUP DEALLOCATE command.

- The exceptional condition report consists of:

  --The space-allocation consistency analysis, indicating the consistency of free space and allocated space and noting any lost free space, overlapped extents, unspared defective sectors, or free extents out of order in the free-space table

    Extent overlaps (doubly allocated pages) indicate a hardware or software error. These should be reported to your Tandem representative.

    Lost free space is a consequence of a total system crash or a double CPU failure.

  --The summary of space allocation anomalies, which summarizes the space-allocation consistency analysis above into the categories of disc free-space table inconsistencies, doubly allocated pages, and lost free-space pages

--The media failure analysis, which reports on whether the disc (primary or mirror) is down or has a given number of unspared defective sectors. If the disc has a number of unspared defective sectors, DSAP displays the message:

```
{ Primary } disc has <num> unspared defective sectors.
{ Mirror  }
Please eliminate bad sectors using the PUP SPARE command.
```

Figure 8-1 shows a summary report for a full-volume analysis. The command to show this report is:

```
:DSAP $SYSTEM
```

```
Disc Space Analysis Program (DSAP) -- T9074B00 -- (18MAR85) --  5/5/85 13:30:41
Volume $SYSTEM is logical device 5
Device type is 3, subtype 3 ( 4104 -- 240MB )

     114,026 pages (2048 bytes) on volume
 233,525,248 bytes on volume


Summary of space use on $SYSTEM

      23,061 free pages in 354 extents (20.2%).

      90,296 allocated pages in 1,199 files in 1,922 extents (79.1%).

       5,395 unused pages in 269 files (4.7%).

       3,017 deallocatable extent pages in 22 files (2.6%).


Space Allocation Consistency Analysis:

     1 lost free space pages (@page 1657)
     1 doubly-allocated pages (@page  1665),
                              files: $SYSTEM.SALEFRC.JOE
                                     $SYSTEM.SUPPORT.SAM
     8 lost free space pages (@page 6978)
     8 lost free space pages (@page 12871)

Summary of space allocation anomalies:

         0 disc free space table inconsistencies.
         1 doubly-allocated pages.
        17 lost free space pages.


Media Failure Analysis:

Primary disc is down.
Mirror disc has no unspared defective sector(s).
```

Figure 8-1. Sample Summary Report for an Entire Disc

When DSAP performs a summary analysis for a single user, the summary report is truncated, as illustrated in Figure 8-2. The command to show this report is:

```
:DSAP $DATA, USER MFG.NIMROD
```

or

```
:DSAP $DATA, USER 3,85
```

```
Disc Space Analysis Program (DSAP) -- T9074B00 -- (18MAR85) --  4/05/85 12:51:52
Summary of space use for MFG.NIMROD on $DATA
   3,898 allocated pages in 311 files in 833 extents (3.4%).
     504 unused pages in 117 files (0.4%).
      14 deallocatable extent pages in 6 files (0.0%).
```

Figure 8-2. Sample Summary Report for a Single User

The percentages given in parentheses for the summary of space use are percentages of the total disc pages on the volume. Note in Figure 8-1 that the percentage of allocated pages added to the percentage of free pages is not exactly equal to 100 percent. The space representing that difference is used by the system tables (containing the disc directory, free-space table, spare tracks table, volume label, and others).

DSAP Space Distribution Reports

DSAP performs space-distribution analysis for the following types of space shown in Table 8-2:

Table 8-2. Types of Space-Distribution Analysis

Type of Space	Report Option
Free-space extents	FREESPACE
Allocated file extents	FILESPACE
Both free and allocated extents	SPACE
Entire files	FILESIZE

All report types first show the summary report, as in Figure 8-1.

Figure 8-3 shows the free-space distribution report produced by the following command:

```
:DSAP $<volume> FREESPACE
```

For each extent size, the free-space distribution report displays:

1. The number of free extents
2. The total pages (of 2048 bytes each) occupied by those free extents
3. The total pages (of 2048 bytes each) occupied by those free extents of that size or smaller
4. The free space used as a percentage of all that disc's free space
5. The free space used as a percentage of all disc space

Free Space Distribution

------Free Extents--------				---Free Space---		---Disc Space----	
Extent Size	Count	Pages	Cumulative Pages	--Distribution-- Space	Cumulative	--Distribution--- Space	Cumulative
1	126	126	126	0.4%	0.4%	0.1%	0.1%
2	22	44	170	0.1%	0.6%	0.0%	0.1%
3	3	9	179	0.0%	0.6%	0.0%	0.1%
4	6	24	203	0.0%	0.7%	0.0%	0.1%
5	2	10	213	0.0%	0.8%	0.0%	0.1%
6	3	18	231	0.0%	0.8%	0.0%	0.1%
7	1	7	238	0.0%	0.9%	0.0%	0.1%
8	4	32	270	0.1%	1.0%	0.0%	0.2%
9	1	9	279	0.0%	1.0%	0.0%	0.2%
10	2	20	299	0.0%	1.1%	0.0%	0.2%
11	1	11	310	0.0%	1.1%	0.0%	0.2%
17	1	17	327	0.0%	1.2%	0.0%	0.2%
19	1	19	346	0.0%	1.3%	0.0%	0.2%
20	1	20	366	0.0%	1.3%	0.0%	0.3%
22	1	22	388	0.0%	1.4%	0.0%	0.3%
25	1	25	413	0.0%	1.5%	0.0%	0.3%
37	1	37	450	0.1%	1.7%	0.0%	0.3%
39	1	39	489	0.1%	1.8%	0.0%	0.4%
62	1	62	551	0.2%	2.0%	0.0%	0.4%
257	1	257	808	0.9%	3.0%	0.2%	0.6%
276	1	276	1084	1.0%	4.1%	0.2%	0.9%
1600	1	1600	2684	6.0%	10.1%	1.3%	2.2%
23694	1	23694	26378	89.8%	100.0%	19.6%	21.9%

Figure 8-3. Sample Free-Space Distribution Report

Figures 8-4 and 8-5 show the file extent-size distribution report and file-size distribution report produced by these commands:

```
:DSAP $<volume> FILESPACE
```

and

```
:DSAP $<volume> FILESIZE
```

These reports display information in the same manner as the free-space distribution report.

File Extent Size Distribution

--Allocated File Extents--				---File Space---		---Disc Space----	
Extent			Cumulative	--Distribution--		--Distribution---	
Size	Count	Pages	Pages	Space	Cumulative	Space	Cumulative
1	4293	4293	4293	4.6%	4.6%	3.5%	3.5%
2	5535	11070	15363	12.0%	16.6%	9.2%	12.7%
3	1023	3069	18432	3.3%	20.0%	2.5%	15.3%
4	1650	6600	25032	7.1%	27.1%	5.4%	20.8%
5	687	3435	28467	3.7%	30.9%	2.8%	23.6%
6	386	2316	30783	2.5%	33.4%	1.9%	25.5%
7	339	2373	33156	2.5%	36.0%	1.9%	27.5%
8	893	7144	40300	7.7%	43.7%	5.9%	33.4%
9	135	1215	41515	1.3%	45.0%	1.0%	34.5%
10	374	3740	45255	4.0%	49.1%	3.1%	37.6%
11	81	891	46146	0.9%	50.1%	0.7%	38.3%
12	112	1344	47490	1.4%	51.5%	1.1%	39.4%
13	72	936	48426	1.0%	52.6%	0.7%	40.2%
14	50	700	49126	0.7%	53.3%	0.5%	40.8%
15	24	360	49486	0.3%	53.7%	0.2%	41.1%
16	318	5088	54574	5.5%	59.2%	4.2%	45.3%
17	16	272	54846	0.2%	59.5%	0.2%	45.5%
				.		.	
				.		.	
				.		.	
77	1	77	79110	0.0%	85.9%	0.0%	65.7%
81	1	81	79191	0.0%	86.0%	0.0%	65.8%
83	2	166	79357	0.1%	86.2%	0.1%	65.9%
84	1	84	79441	0.0%	86.2%	0.0%	66.0%
91	1	91	79532	0.0%	86.3%	0.0%	66.1%
92	1	92	79624	0.0%	86.4%	0.0%	66.1%
95	3	285	79909	0.3%	86.8%	0.2%	66.4%
96	3	288	80197	0.3%	87.1%	0.2%	66.6%
97	2	194	80391	0.2%	87.3%	0.1%	66.8%
100	18	1800	82191	1.9%	89.2%	1.4%	68.3%
101	3	303	82494	0.3%	89.6%	0.2%	68.5%
106	1	106	82600	0.1%	89.7%	0.0%	68.6%
108	3	324	82924	0.3%	90.0%	0.2%	68.9%
127	20	2540	85464	2.7%	92.8%	2.1%	71.0%
128	1	128	85592	0.1%	92.9%	0.1%	71.1%
200	4	800	86392	0.8%	93.8%	0.6%	71.8%
381	3	1143	87535	1.2%	95.0%	0.9%	72.7%
506	2	1012	88547	1.0%	96.1%	0.8%	73.5%
513	1	513	89060	0.5%	96.7%	0.4%	74.0%
625	1	625	89685	0.6%	97.4%	0.5%	74.5%
1000	1	1000	90685	1.0%	98.5%	0.8%	75.3%

Figure 8-4. Sample File Extent-Size Distribution Report

File Size Distribution

-----Total File Size------				---File Space---		---Disc Space----	
File Size	Count	Pages	Cumulative Pages	--Distribution-- Space	Cumulative	--Distribution--- Space	Cumulative
0	117	0	0	0.0%	0.0%	0.0%	0.0%
1	935	935	935	1.0%	1.0%	0.7%	0.7%
2	1405	2810	3745	3.0%	4.0%	2.3%	3.1%
3	188	564	4309	0.6%	4.6%	0.4%	3.5%
4	937	3748	8057	4.0%	8.7%	3.1%	6.6%
5	83	415	8472	0.4%	9.2%	0.3%	7.0%
6	300	1800	10272	1.9%	11.1%	1.4%	8.5%
7	46	322	10594	0.3%	11.5%	0.2%	8.8%
8	293	2344	12938	2.5%	14.0%	1.9%	10.7%
9	144	1296	14234	1.4%	15.4%	1.0%	11.8%
10	208	2080	16314	2.2%	17.7%	1.7%	13.5%
11	13	143	16457	0.1%	17.8%	0.1%	13.6%
12	196	2352	18809	2.5%	20.4%	1.9%	15.6%
13	16	208	19017	0.2%	20.6%	0.1%	15.8%
14	68	952	19969	1.0%	21.6%	0.7%	16.5%
15	47	705	20674	0.7%	22.4%	0.5%	17.1%
16	134	2144	22818	2.3%	24.7%	1.7%	18.9%
				.			.
				.			.
				.			.
367	2	734	77720	0.7%	84.4%	0.6%	64.6%
381	3	1143	78863	1.2%	85.6%	0.9%	65.5%
397	1	397	79260	0.4%	86.0%	0.3%	65.8%
400	1	400	79660	0.4%	86.5%	0.3%	66.2%
406	1	406	80066	0.4%	86.9%	0.3%	66.5%
462	2	924	80990	1.0%	87.9%	0.7%	67.3%
500	1	500	81490	0.5%	88.5%	0.4%	67.7%
502	1	502	81992	0.5%	89.0%	0.4%	68.1%
506	2	1012	83004	1.0%	90.1%	0.8%	68.9%
513	1	513	83517	0.5%	90.7%	0.4%	69.4%
538	1	538	84055	0.5%	91.3%	0.4%	69.8%
600	1	600	84655	0.6%	91.9%	0.4%	70.3%
621	3	1863	86518	2.0%	93.9%	1.5%	71.9%
1000	3	3000	89518	3.2%	97.2%	2.4%	74.4%
1016	1	1016	90534	1.1%	98.3%	0.8%	75.2%
1524	1	1524	92058	1.6%	100.0%	1.2%	76.5%

Figure 8-5. Sample File-Size Distribution Report

DSAP Report by Subvolume

The first portion of this report shows the summary report, as in Figure 8-1. Figure 8-6 below shows the subvolume summary report produced by the command:

```
:DSAP $<volume> BYSUBVOL
```

Subvol Summary Report

Subvolume Name	Files	Total Pages	Unused Pages	Dealloc Pages	Large File	Min Age
FREE SPACE		22983				
DISC DIRECTORY		1879				
TEMPORARY FILES	1	6	0	0	6	0
ALPHA	25	912	229	156	400	1
AN123X	15	229	18	0	56	22
AN456Y	17	305	20	0	56	22
ANSERV	3	276	11	0	148	502
AS181	1	196	9	0	196	44
AS262A	1	2	0	0	2	33
ATCHISON	4	234	4	0	212	28
AUDIT1	86	1713	1051	0	100	44
AUDIT2	4	62	5	0	40	269
AUDITFIN	37	551	304	6	100	348
AUDIT	1	10	1	0	10	106
AXLE	1	4	3	0	4	203
AXLE2	35	541	38	0	116	23
B02DICT	7	105	63	48	50	351
BALANCE	92	1169	171	2	130	28
BASEA05	69	2779	261	0	265	50
BLAST	23	141	35	0	52	6
BLAST1	12	175	10	0	44	149
BONDS	6	15	1	0	6	13
BUDGET	11	73	17	0	36	42
BUDGET84	3	55	9	0	25	54
C232A00	1	14	0	0	14	365
CASES	27	213	40	0	88	167

Figure 8-6. Sample Subvolume Summary Report

The first two entries of the subvolume summary report portion always list the number of pages of free space available, followed by the number of pages devoted to the disc directory. The next entry is for temporary files, if any exist. The remaining entries are an alphabetic list of subvolumes, displaying for each:

1. The number of files
2. The total pages allocated
3. The total pages of unused space
4. The pages of unused space in deallocatable extents
5. The pages in the largest file
6. The age (in days) of the most recently modified file, called the minimum age

DSAP Report by User

The first portion of this report shows the summary report as in Figure 8-1. Figure 8-7 shows the user summary report portion produced by this command:

```
:DSAP $<volume> BYUSER
```

User Summary Report

User Name	ID	Files	Total Pages	Unused Pages	Dealloc Pages	Large File	Min Age
FREE SPACE			22983				
DISC DIRECTORY			1879				
SALEFRC.JOE	1,1	294	3857	869	561	1024	0
SALEFRC.NANCY	1,2	126	2845	299	4	371	0
SALEFRC.DAVID	1,3	6	290	9	0	82	75
SALEFRC.ANNE	1,9	1	4	3	0	4	203
SALEFRC.MIKE	1,17	1	4	0	0	4	578
SALEFRC.GLENN	1,18	63	728	54	1	175	0
SALEFRC.PENNY	1,34	89	611	94	0	50	6
SALEFRC.LOU	1,36	1	0	0	0	0	43
.		.			.		
.		.			.		
.		.			.		
SALEFRC.BARB	1,143	1	196	9	0	196	44
SALEFRC.????????	1,153	67	907	84	0	350	162
SALEFRC.CHARLIE	1,156	7	19	0	0	6	36
SALEFRC.KATE	1,162	2	8	4	0	4	71
SALEFRC.ANDY	1,163	157	2193	1107	84	100	1
SALEFRC.DENNIS	1,172	116	3361	555	2	210	0
SALEFRC.RICH	1,180	147	3913	395	20	375	0
SALEFRC.EVE	1,199	53	509	66	0	56	0
SALEFRC.PDT	1,201	11	416	5	0	81	0
SALEFRC.COMM	1,203	11	184	3	0	80	30
SALEFRC.ROBERT	1,207	37	1029	185	0	192	162
SALEFRC.TCM	1,253	1	240	5	0	240	22
????????.??????	7,255	1	14	0	0	14	365
SUPPORT.SAM	20,10	24	331	31	2	60	0
CPU.MANAGER	143,255	14	25	3	0	5	504
SUPER.RELEASE	255,0	2	60	10	0	30	125
SUPER.SUPER	255,255	60	2985	1216	190	400	5

Figure 8-7. Sample User Summary Report

For each user ID, the report by user displays:

1. The number of files
2. The total pages allocated
3. The pages of unused space
4. The pages in deallocatable extents
5. The pages in the largest file
6. The age (in days) of the last modified file (minimum age)

Note that the report shows user IDs (1,153 and 7,255) that have files allocated on the volume but are not known to the system. This occurs if users are deleted from the system, but the files they own are not deleted.

DSAP Detail Report

The first portion of this report gives the summary report, as in Figure 8-1. Figure 8-8 shows the user detail report generated by this command:

```
:DSAP $DATA, DETAIL, DEALLOC 1, UNUSED 50
```

```
User Detail Report

Selection Criteria: total unused pages >= 50
                    pages in unused extents >= 1
User                                    Total Unused Dealloc
Name/ID    Filename   Type Code  Pages  Pages  Pages  Exts  Age

SALEFRC.TOM
(1,1)    TOM.WORKFILE        101    10     10     10     1     0

SALEFRC.DICK
(1,108)  RHC.GIXIX           101     4      2      2     2     8
         RHC.FORM            101     4      2      2     2     9
         RHC.START           100L    4      2      2     2     9
         RHC.FACTOR          101    18      4      4     9     7
         RHC.SALIST     ?    110     4      2      2     2     9

SALEFRC.HARRY
(1,163)  ASREPORT.CPU        100I   48      9      8     6     7
         ASGAUGE.KDATA  K    3000A  18      5      4     9   167
         ASGAUGE.DATA   RP          30      2      2    15   188
```

Figure 8-8. Sample of the File Detail Report

The user detail report displays a detailed list of all, or a selected subset, of the files on a disc. As indicated in the command syntax description, many options are available to control the specific contents of this report.

For each file, the report displays the following information:

1. The file type:

blank	Unstructured	
E	Entry-sequenced	
K	Key-sequenced	
R	Relative	
P	Partitioned	Independent of above
?	Crash-opened	Independent of above

   For example, a crash-opened relative file that is partitioned is represented by:

   ?RP

2. Alongside the file code are indications if a file is:

A	Audited
L	Licensed
I	PROGID (program ID) option set

Refer in Section 2 to the FUP DUP command for more information on file codes.

FORMAT OF THE PERMANENT WORK FILE

The DDL description of the permanent work file (explained under "DSAP Considerations") is contained in a file that resides in $SYSTEM.SYSTEM.DSAPDDL, which is released with each version of DSAP. You should always use the current DSAPDDL to analyze the work file. Refer to the Data Definition Language (DDL) Reference Manual for more information on DDL.

Figure 8-9 illustrates the format of this DSAPDDL template that is used by ENFORM to produce written reports.

```
record directory-record.
   file is assigned unstructured.
   02 fnam.
      03 volume          type binary occurs 4 times .
      03 subvol          type binary occurs 4 times .
      03 filename        type binary occurs 4 times .

   02 file              redefines fnam .
      03 volume          pic "x" occurs 8 times.
      03 subvol          pic "x" occurs 8 times.
      03 filename        pic "x" occurs 8 times.

*                           owner's group/user id
   02 owner             type binary .

   02 owning            redefines owner .
      03 group           type binary 8 unsigned .
      03 user            type binary 8 unsigned .

*                        FCBPROTECTION and decoded protection
   02 fcbprotection     type binary unsigned .
   02 security .
      03 read            pic "x" .
      03 write           pic "x" .
      03 execute         pic "x" .
      03 purge           pic "x" .
```

Figure 8-9. DDL Format of the Permanent Work File
(Continued on next page)

```
*                                File code
   02 filecode              type binary unsigned .
*                                FCBFLAGS
*                                FCBFLAGS.<13:15> = filetype
   02 fcbflags              type binary unsigned .
*                                File type: U, E, R, K
   02 filetype              pic "x" .
*                                Audited file: "A"
   02 audited               pic "x" .
*                                Licensed file: "L"
   02 licensed              pic "x" .
*                                Partitioned file: "P"
   02 partitioned           pic "x" .
*                                Crash-opened file: "?"
   02 crashopened           pic "x" .
*                                File with PROGID set: "I"
   02 progid                pic "x" .
*                                timestamp of latest modification.
   02 lastmod               type character 6 .
   02 modtime               type binary occurs 3 times
                            redefines lastmod.
*                                Number of extents
   02 numexts               type binary .
*                                Primary extent size
   02 primary               type binary .
*                                Secondary extent size
   02 secondary             type binary .
*                                File size in pages
   02 filesize              type binary 32 .
*                                Byte address of eof
   02 eof                   type binary 32 .
*                                Pages in extents past the eof
   02 dealloc               type binary 32 .
   end
```

Figure 8-9. DDL Format of the Permanent Work File (Continued)

If the entire disc is analyzed, the first two file records are named "****FREE.SPACE" and "****DISC.DIRECTRY". These records indicate, respectively, the amount of free space and the size of overhead space (directory, free-space table, and so forth) on the disc. If a group or a user is analyzed, only the files of the group or user appear in the work file.

The order of the file records depends on the report options selected. The BYUSERS and DETAIL reports sort the records by the owner ID; otherwise, the records are sorted by the subvolume and file name.

To run an ENFORM query against the work file, use the following procedure and assume you want a list of all EDIT files (and the corresponding end-of-file markers) owned by user ID 1,100:

1. Create an EDIT file for ENFORM to use as a source. The file should have:

```
?DICTIONARY
?ASSIGN Directory-Record TO <workfile-name>
OPEN Directory-Record;
LIST File, EOF WHERE (FileCode = 101)   AND
                     (Owning.Group = 1) AND (Owning.User = 100);
CLOSE Directory-Record;
```

2. Then run:

```
ENFORM / IN <edit-file>, OUT <outfile> /
```

Alternatively, you can run ENFORM interactively, entering the commands listed above in the same sequence.

DSAP EXAMPLES

The following examples illustrate typical uses at a variety of user levels, from system manager to group manager to individual user:

1. To obtain a hard copy of all DSAP analysis reports for the volume $DATA, including the list of all user files that the detail report provides, a system manager or system operator enters:

```
:DSAP /OUT $S.#PRTR, NOWAIT/ $DATA ANALYSIS DETAIL
```

2. To track space use and to detect errors on $DATA before they cause problems, a system manager or system operator enters the following command to get the free-space distribution report on $DATA:

```
:DSAP $DATA, EXTENTCHECK, FREESPACE
```

   Obtaining this report on a daily basis and examining the space-allocation consistency analysis portion of it serves as a sort of health check of the disc.

3. To obtain a report for distribution to each user, describing the files allocated to each of those users on the volume $DATA, a system manager or system operator enters:

```
:DSAP $DATA, DETAIL, SEPARATE
```

4. To obtain a report for distribution to each user in a group, describing the files allocated to each of those users on the volume $DATA, the group manager enters:

```
:DSAP $DATA, USER ADMIN.*, SEPARATE, DETAIL
```

5. For general security checking (to detect programs that could be used to breach security), a system manager or system operator enters:

```
:DSAP $DATA, DETAIL, LICENSED, PROGID
```

6. To obtain a listing of all files on the volume $DATA that are owned by the individual user in the sales group named Janet, enter:

```
:DSAP $DATA, USER sales.janet, DETAIL
```

7. To obtain only the summary report and a list of all your files on the volume $DATA, enter:

```
:DSAP $DATA, USER, DETAIL
```

DCOM--THE SPACE COMPRESSOR

DCOM analyzes the current space allocation on a disc and then moves extents from one place to another.

- DCOM reduces the number of free-space extents, thus reducing the effort required to allocate or deallocate space
- DCOM combines free space into larger extents, thus allowing the allocation of files with larger extents and decreasing the incidence of GUARDIAN file-system error 43 (UNABLE TO OBTAIN DISC SPACE FOR EXTENT)

Refer to the System Operator's Guide for information on how DCOM works. DCOM error messages can be found in the System Messages Manual.

Like DSAP, DCOM is a noninteractive program. By default, it displays the preliminary reports and compression messages at the home terminal. Or, you can designate any other valid output file in the OUT parameter of the startup message.

DCOM Program

The syntax of the command to run DCOM is:

```
DCOM  [ / <run-options> / ] [ $<volume> | HELP ]
      [ MAXMOVES <integer>   ]
      [ WORKFILE $<volume>   ]    ...
      [ IGNOREBADSECTORS     ]
      [ VERIFY               ]
```

<run-options>

are options to the GUARDIAN command interpreter RUN command, as described in Section 2. In particular, if you wish output to be sent to a file other than the home terminal, specify the OUT option.

$<volume>

specifies the disc volume on which the compression is to be performed. Optional keywords and parameters can follow the $<volume> specification in any order. If you omit $<volume>, a brief description of the DCOM program and its options is displayed (identical to that displayed if you specify HELP).

HELP

displays a brief description of the DCOM program and its options.

MAXMOVES <integer>

limits the total number of extents that DCOM can move during any one compression. The default limit is 32768.

WORKFILE $<volume>

specifies a volume for the allocation of a temporary file. If a specific disc is close to being full, and DCOM attempts to map the extended data segment to a temporary file on that disc, DCOM may terminate with GUARDIAN file system error 43 (NO DISC SPACE). You can, therefore, specify a disc that is not so full as the work-file volume in order to avoid this problem.

If you do not specify the WORKFILE option, the DCOM work file is allocated, by default, on the volume where the program swap file resides. (You can alter this default value using BINDER. See "Altering the Default Work File Volume" at the end of this section.) Unlike DSAP, DCOM does not recognize a permanent WORKFILE option since the compression process changes the work file to an unusable form.

IGNOREBADSECTORS

specifies that DCOM is to perform the disc compression even if it detects, during the initial analysis, that the disc contains unspared bad sectors. By default, DCOM terminates in this event, producing the following error message:

Please eliminate bad sectors using the PUP SPARE command.

VERIFY

specifies that every write operation DCOM performs to the disc is to be verified. (This option causes DCOM to run considerably longer.)

DCOM Considerations

- DCOM compresses an online disc. That is, you can run it in parallel with other processing without removing the affected disc from service. DCOM is not, however, totally invisible to other processing. Execution of DCOM can have three main side effects:

  --While DCOM is moving a file extent, it must have exclusive access to the file. Naturally, a file already open is not affected. If, however, an important application will fail if it cannot open some file, then DCOM should not be run on the affected disc when that application is running. Although the probability of conflict is small, it does exist.

  --Since the predominant activity of DCOM is disc I/O, DCOM can negatively affect an application that is disc-I/O bound. Running DCOM at a low priority reduces its effect on other processing but does not eliminate it.

  --DCOM moves individual file extents from one disc address to another. If a file or group of files have been allocated in contiguous extents to reduce the effect of seeking, DCOM may make the extents noncontiguous.

- Since DCOM is designed for online discs, it does not perform perfect compression. You can sometimes achieve additional

compression by running DCOM several times serially, but you should exercise discretion in doing this. The starting overhead of each run is significant (because it requires a read and sort of the disc directory), and DCOM is unable to determine when additional runs are pointless.

- To prevent two or more DCOM processes from running simultaneously (which can actually increase rather than reduce disc fragmentation), DCOM has a mutual exclusion feature. When it starts compressing a disc, DCOM attempts to create and open a file called SYSDCOM.RECOVERY (secured "NNNN"). If the file already exists, DCOM tries to open it exclusively. If exclusive access cannot be obtained, DCOM terminates with the following error message:

  ```
  Unable to open SYSDCOM.RECOVERY file, error: 12
  ```

  This refers to GUARDIAN file-system error 12 (FILE IN USE). When the compression is complete, DCOM closes and purges the file.

- If DCOM happens to be running when a system crash occurs, temporary files created by DCOM may remain on the disc under compression. These files are owned by the super ID (255,255) and secured "---N". Such files also have the CLEARONPURGE option set if they are being used to move the extents of files set for CLEARONPURGE. (The CLEARONPURGE option is set by the FUP SECURE command, as described in Section 2.)

- DCOM has no internal security restrictions other than the fact that it requires licensing. The system manager controls access to DCOM using the standard disc file security scheme.

- In order to ensure compatibility with low-level system interfaces, DCOM requires that its version match the version of the GUARDIAN operating system.

- You can exit a running DCOM process at any time by pressing the BREAK key. DCOM continues to run in the background until it is finished. If DCOM is sending output to your terminal, you can resume running DCOM by typing "PAUSE".

- To stop a running DCOM process, press the BREAK key and, when the COMINT prompt (:) appears, issue a STOP command. Terminating the DCOM program in the midst of compression simply leaves the disc in a partially compressed state, which is still an improvement over a totally uncompressed disc. Early termination of DCOM has no detrimental effects.

- Using the MAXMOVES option, you can perform a fixed amount of compression each day, depending on the amount of quiet time a given system has (when no performance-critical applications are running).

EXAMPLE OF COMPRESSION OUTPUT

In its preliminary stage, DCOM produces the same summary report (Figure 8-1) and space-distribution reports (Figure 8-3, 8-4, and 8-5) produced by DSAP.

Before the Move

When the compression phase begins, but before DCOM moves any extents, messages are written to the operator console in the format illustrated in Figure 8-10.

```
DCOM: Starting disc compression for $DATA.
DCOM:     23199 pages of free space in 356 extents (20.3%).
DCOM:      1332 pages in largest free extent.
```

Figure 8-10. Format of DCOM Report (Before the Move)

During the Move

For each file extent that it moves, DCOM sends a report to the current OUT <listfile>, noting each move DCOM makes and which files were directly affected. As shown in Figure 8-11 below, in the fifth move, DCOM moves extent number 11 of the file XRAYENF.QCACHE from page 52181 on the disc to its new location at page 5110 on the disc at the time given. (This file was created with a primary extent size of one page and a secondary extent size of one page.)

Move Num	To Page-Num	From Page-Num	Extent Size	Extent Num	File Name (Pri, Sec)	Time
1	3424	52710	1	0	XRAYENF.ZQTXRATE (1,1)	12:30:16
2	3565	52705	1	1	XRAYENF.ZQCPU (1,1)	12:30:18
3	4794	52704	1	0	XRAYENF.ZQCPU (1,1)	12:30:19
4	5089	52438	1	0	XRAYENF.TRANRATE (1,1)	12:30:20
5	5110	52181	1	11	XRAYENF.QCACHE (1,1)	12:30:22
6	5184	52180	1	0	XRAYENF.QCACHE (1,1)	12:30:23
7	5189	52179	1	1	XRAYENF.QBALANC2 (1,1)	12:30:24
8	5208	52178	1	0	XRAYENF.QBALANC2 (1,1)	12:30:26
9	5210	51973	1	1	XRAYENF.OQP (1,1)	12:30:27
10	5221	51972	2	2	XRAYENF.OQP (2,2)	12:30:29

Figure 8-11. Format of DCOM Report (During the Move)

After the Move

When DCOM finishes moving extents, it produces another report (see Figures 8-12). The first line tells you the total number of extents and pages moved.

The second line gives the new number of extents, which are fewer and larger than before the move, depending on the success of the compression.

DCOM produces much the same information on a disc as the PUP LISTFREE HIST command, but DCOM does it faster. A LISTFREE HIST would show a number of fragments equal to the extents DCOM shows, unless there was allocation or deallocation file activity while DCOM was running. LISTFREE also gives the address of each fragment on the disc in page number form. The total number of pages of free space (and the corresponding percentage) would be unchanged.

The third line lists the number of pages in the largest free extent on the disc; to see the rest, refer to the space distribution reports that DCOM also produces at this time.

```
  227 extents moved, 1019 pages.
23199 pages of free space in 147 extents (20.3%).
 1402 pages in largest free extent.
```

Figure 8-12. Format of DCOM Report (After the Move)

When the compression is complete, messages are written to the operator console, as shown in Figure 8-13.

```
DCOM: Terminating disc compression for $DATA.
DCOM:         227 extents moved, 1019 pages.
DCOM:       23199 pages of free space in 147 extents (20.3%).
DCOM:        1402 pages in largest free extent.
```

Figure 8-13. Format of DCOM Report (Concluded)

How long a single compression run takes can vary considerably, depending on the size of the disc and the degree of fragmentation on it. Subsequent compressions should take less time than the first.

ALTERING THE DEFAULT WORK-FILE VOLUME

Both DSAP and DCOM allow the system manager to use BINDER to alter the default work volume specification. The following example illustrates the brief BINDER session required to accomplish the change. (In the DSAP and DCOM object files, the volume name of the default work file is located at the beginning of the procedure SELECT^WORK^VOLUME.)

In this example, you are setting the volume for the DSAP work file to $TEMP. To accomplish the same for DCOM, simply substitute DCOM wherever DSAP currently appears. Where the characters for the volume name $TEMP appear, substitute (by twos in quotes: "$T", "EM", "P ", " ") the characters in the name of your volume:

```
:VOLUME $SYSTEM.SYSTEM
:BIND
@FILE DSAP
@ADD *
@MODIFY CODE SELECT^WORK^VOLUME ASCII 0, "$T","EM","P "," "
@BUILD DSAP!
@EXIT
:FUP LICENSE DSAP
```

The FUP LICENSE command is optional, depending on what kind of file security you desire. It is necessary to relicense the program because the BINDER BUILD command removes the old license. Refer to the BINDER Manual for complete information about BINDER, and to Section 2 for details about the FUP LICENSE command.

SECTION 9

OTHER UTILITY PROGRAMS

This section presents the command syntax and considerations for five utility programs associated with the GUARDIAN operating system. The following programs are described:

DELAY	DIVER	ERROR
PEEK	SPOOL	

- DIVER and DELAY test programs that run as NonStop process pairs. Such programs should continue to execute in the event of a processor failure. DIVER causes a processor to fail and then makes the processor ready for a reload. Used with the DELAY program, DIVER causes repeated failures and reloads of the processors in a system. Thus, DIVER and DELAY demonstrate that application processes can run while processors are halted and reloaded. (DIVER is reserved for use by the system operator.)

- ERROR displays the error message associated with the number of a file-system error.

- PEEK is a privileged utility program that reports statistical information internally maintained by the operating system within the processor module. You use PEEK to monitor processor activity for system storage pools, paging activity, send instructions, and interrupt conditions.

- SPOOL is a command used by system operators to run the spooler supervisor when cold or warm starting the spooler.

PROGRAM DESCRIPTIONS

The rest of this section contains descriptions of the syntax for the five utilities listed in the previous subsection. Each program description includes:

- A summary of the program's function
- The command syntax you use to start a process, including a description of each parameter and variable
- The listing format used for program ouput (for those programs that produce a display or listing output)
- Considerations for use of the program
- Examples showing how to use the program

DELAY Program

You use the DELAY program to delay execution of the COMINT program for the amount of time you specify in your DELAY command. The DELAY process calls the DELAY operating system procedure for the number of "ticks" (hundredths of a second), seconds, or minutes you specify. Then DELAY stops, thus reawakening the COMINT process that started it.

The command to run the DELAY program is:

DELAY / <run-option> [, <run-option>]... / <integer> { TIC | SEC | MIN }

DELAY

is an implicit RUN command that starts a DELAY process.

<run-option>

is any valid <run-option> for the RUN command in COMINT. (See the syntax for the RUN command in Section 2.)

<integer>

is the number of TICs (hundreths of a second), SECs (seconds), or MINs (minutes) to delay.

TIC, SEC, or MIN

specifies the unit of measurement for <integer>. (See the description of the <integer> option.)

Example

The following example uses DIVER and DELAY to automatically fail and reload both processors of a two-processor system. This example causes a processor failure cycle about once a minute. You might want to increase the amount of time in the cycle to allow proper recovery by your applications. You should have at least one COMINT program running as a NonStop process pair before starting the cycle.

First, create two files in the format of the EDIT program and name them DIVE0 and DIVE1. The files should contain the COMINT commands shown here.

The contents of DIVE0 are:

```
DIVER / CPU 1 /
DELAY 30 SEC
RELOAD 1
DELAY 30 SEC
COMINT/ CPU 1, NAME $DIVE1, IN DIVE1 /
```

The contents of DIVE1 are:

```
DIVER / CPU 0 /
DELAY 30 SEC
RELOAD 0
DELAY 30 SEC
COMINT/ CPU 0, NAME $DIVE0, IN DIVE0 /
```

To start the failure cycle, start a COMINT process using one of these files:

```
:COMINT /CPU 0, NAME $DIVE0, IN DIVE0, OUT $0 /
```

To stop the failure cycle, enter:

```
:STOP $DIVE0
:STOP $DIVE1
```

DIVER Program

You run the DIVER program in a processor you want to "fail." DIVER takes over the processor and stops all activity in it. The other processors in the system consider that this processor has failed. After 15 to 30 seconds, DIVER sets up the processor so that it is ready for a RELOAD command (just as if the RESET and LOAD buttons on the processor had been pushed). The processor lights show %70707 for a few seconds and then show %0. The RUN lamp is off at this point. Next, the system operator can start a RELOAD process for this processor at any time.

The commmand to run the DIVER program is:

```
DIVER / <run-option> [ , <run-option> ]... /

DIVER

   is an implicit RUN command that starts a DIVER process.

<run-option>

   is any valid <run-option> for the RUN command in COMINT.
   (See the syntax for the RUN command in Section 2.)
```

Consideration

Before it downs a processor, DIVER checks two things: First, it verifies that its process accessor ID belongs to group 255; that is, that the program was run by a user with system operator capability. Second, DIVER verifies that bit 0 of the processor's switch register is on, since this is necessary for reloading the processor. If either condition is not met, the program abends.

Example

This example uses the CPU <cpu-number> option to start a DIVER process in CPU5. DIVER then downs CPU 5:

```
DIVER / CPU 5 /
```

ERROR Program

The ERROR program displays the error message associated with a file-system error number.

The command to run the ERROR program is:

```
ERROR [ / <run-option> [ , <run-option> ]... / ]
   <error-number>
```

ERROR

is an implicit RUN command that starts an ERROR process.

<run-option>

is any <run-option> for the RUN command in COMINT. (See the syntax of the RUN command in Section 2.) One commonly used <run-option> is:

OUT <list-file>

names any device, process, or $RECEIVE to receive the listing output from the ERROR program.

<error-number>

is a file-system error code. If you specify -1, ERROR lists all errors.

Consideration

If you specify $RECEIVE as the list file, the creator process can read the error message by doing the following:

```
buf ':=' "$SYSTEM SYSTEM  ERROR   ";
pid ':=' " " & pid FOR 11;
CALL NEWPROCESS(buf,,,,pid,error);  ! create the ERROR
                                    ! process
CALL OPEN(pid,pfnum);               ! open the process.

! format and send the startup message
buf      :=  -1;
buf[21] ':=' ["$RECEIVE", 8 * ["  "]];
buf[33]  :=  error-number-in-binary;
buf[34]  :=  0;
CALL WRITE(pfnum,buf,70);

! read back the error message; one or more reads.
eof := 0;
WHILE NOT eof DO
  BEGIN
    CALL WRITEREAD(pfnum,buf,0,132,countread);
    IF > THEN eof := 1
    ELSE
      BEGIN
           ! process the message
        END;
     END;
```

Example

Suppose you are using COMINT interactively, and you receive this file-system error:

```
PROGRAM FILE ERROR 011
```

You can display the error message that explains file-system error 011 by entering:

```
:ERROR 11
011 file not in directory or record not in file
```

PEEK Program

The PEEK utility program reports statistical information concerned with resource use in the processor.

PEEK can output the block of information one or more times, delaying between outputs as specified by the user. The output can be directed to any standard file, such as the home terminal, a line printer, or an existing disc file, including an EDIT file.

An explanation of how system operators can use this privileged program appears in the System Operator's Guide.

The syntax of the command to run PEEK is:

```
PEEK [ / <run-option> [ , <run-option> ... ] / ]

      [ <samples> [ <delay> ] ]
      [ HELP                  ]
      [ D[YNAMIC]             ]
      [ TIME                  ]
      [ PO[OL]                ]
      [ PA[GING]              ]
      [ MES[SAGES]            ]
      [ INT[ERRUPTS]          ]
      [ ALL                   ]
      [ INIT                  ]
```

<run-option>

is an option to the GUARDIAN command interpreter RUN command, as described in Section 2.

You should include the CPU option to specify the number of the processor module for which you want data; otherwise, a processor is selected arbitrarily. Also include the OUT option if you want output to be sent to a file other than the home terminal.

→

The following parameters can be entered in any order and must be separated from each other by either spaces or commas. If no parameters are selected, information for TIME, POOL, and PAGING is displayed.

<samples>

is the number of times the program is to report the system values. If omitted or 0, one sample is made. The range is 1 to 999.

<delay>

is the amount of time, in seconds, that the program is to pause between successive samples. If omitted or 0, one second is assumed. The range is 1 to 999.

HELP

causes PEEK to display a syntax summary of all PEEK commands.

D[YNAMIC]

specifies that the second and successive samples display only the paging activity that occurs during each sample interval. The first sample displays the information for all the time since the CPU was loaded.

TIME

displays the amount of time the CPU has spent on processes, interrupts, and idle time.

PO[OL]

displays pool management statistics, as further described in the System Operator's Guide.

→

PA[GING]

displays paging statistics, as further described in the System Operator's Guide.

MES[SAGES]

displays the number of unsequenced packets, control packets, and data messages sent by the processor; also displays the distribution of data messages by message length.

INT[ERRUPTS]

displays a count of software interrupts by type, as further described in the System Operator's Guide.

ALL

displays the information produced by the last five options in the order:

TIME + POOL + PAGING + MESSAGES + INTERRUPTS.

INIT

specifies that all pool maximums are reset to equal CURRENT columns in the PEEK report. The MAXIMUM USED for time-list elements, process control blocks, and link control blocks is reset to the CURRENT USED values for these entries. See "Consideration."

Consideration

As a general rule, specify DYNAMIC rather than INIT when you wish to monitor processor activity for a relatively small time period (15 minutes or less). This preserves the measured maximums that are listed in each PEEK report.

Use INIT, on the other hand, if you want to initialize (and thus destroy the past history of) all maximums in the PEEK report. For example, if you run PEEK daily and want a record of each day's activity only, run PEEK in the morning (or when activity is low), and specify INIT. Then run PEEK later during the day, specifying DYNAMIC to preserve the daily maximums. This can help you monitor CPU activity where the level of use fluctuates highly.

Listing Format

Specifying PEEK with no options produces a report equivalent to the reports TIME + POOL + PAGING.

Figure 9-1 shows the format of the PEEK ALL display, which includes the five options:

```
TIME + POOL + PAGING + MESSAGES + INTERRUPTS
```

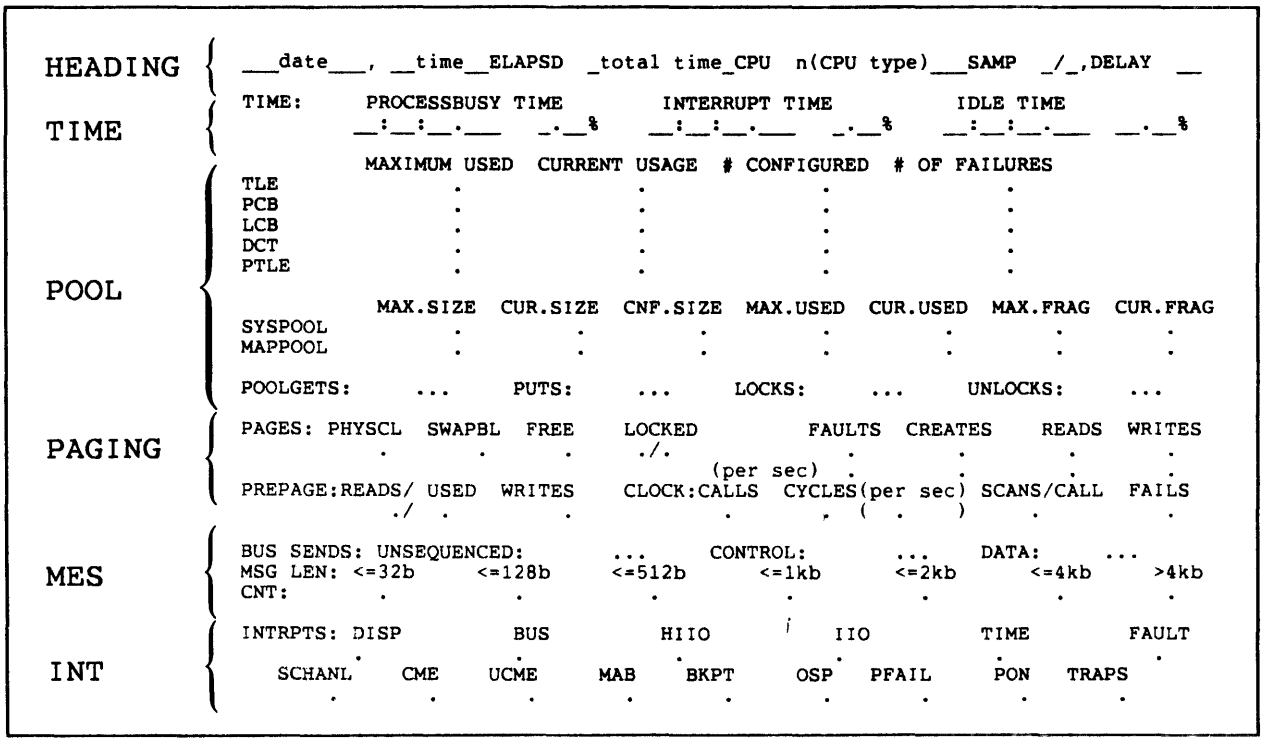


Figure 9-1. PEEK ALL Listing Format

Listing headers for the PEEK display are identified as follows:

Heading Line (Produced by All Options)

ELAPSED	gives the time this display started and the duration on which the sample was based (in hours:minutes:seconds).
CPU	is the number and type (NonStop II or NonStop TXP) of the processor that the statistical information is about.
SAMP DELAY	specifies the number of samples PEEK is to perform and the period of delay between them. If you choose only one sample (the default), this section does not print.

Produced by the TIME Option

PROCESSBUSY TIME	gives the elapsed time the CPU spent executing processes.
INTERRUPT TIME	gives the elapsed time the CPU spent busy with interrupts.
IDLE TIME	gives the elapsed time the CPU spent idle (in hours:minutes:seconds).

Produced by the POOL Option

TLE PCB LCB DCT PTLE	report the state of system tables: time-list elements (TLEs), process control blocks (PCBs), link control blocks (LCBs), destination control table (DCT), and process time-list elements (PTLEs). The PTLE table has no entries when the CPU comes up; entries are added dynamically on a demand basis.
MAXIMUM USED	shows the maximum number of TLEs, PCBs, and LCBs that were ever used.
CURRENT USAGE	shows the number of TLEs, PCBs, and LCBs currently in use.
# CONFIGURED	shows the number of TLEs, PCBs, and LCBs that were defined by SYSGEN parameters. For LCBs, this is the number configured with SYSGEN, minus any that have been reserved by a process calling the RESERVELCBS procedure.
# OF FAILURES	shows the number of allocation failures that occurred for the TLEs and PCBs. A failure count is not kept for LCBs, as each LCB failure is logged on the operator console.
SYSPOOL MAPPOOL	report the state of the system storage pools. Since the pool sizes are dynamic, PEEK displays both maximum and current sizes.

MAX.SIZE
CUR.SIZE
CNF.SIZE

indicate the state of the system storage pools.

MAX = maximum pool size
CUR = current pool size
CNF = configured (absolute maximum) size

Some pool space is usually lost to fragmentation; this loss is reflected in the differences between MAX.SIZE and MAX.USED and between CUR.SIZE and CUR.USED. All counts are in words.

MAX.USED
CUR.USED

show the maximum amount of pool space used and the amount currently in use (in double words).

MAX.FRAG
CUR.FRAG

show the degree of pool fragmentation, as indicated by the discrepancy between the values for SIZE and USED.

POOLGETS

indicates the number of pool allocations for system processes.

PUTS

indicates the number of pool deallocations for system processes.

LOCKS
UNLOCKS

indicate the number of times that memory was locked and unlocked for system processes.

Produced by the PAGING Option

PAGES

indicates the following paging statistics:

PHYSCL is the physical memory size in pages.

SWAPBL is the number of pages not permanently required by the operating system, and thus available for swapping.

FREE is the number of swappable pages not currently assigned to any process.

LOCKED is the number of pages currently locked, divided by the maximum number of swappable pages that can be locked in a processor by various processes at any given time. This number is a fraction of the total number of swap- pable pages in a processor (usually 7/8) and prevents deadlocks due to the entire memory of the processor being locked at any instant.

FAULTS is the total number of page faults that have occurred.

CREATES is the number of faults for which a new page was created without a read operation.

READS is the number of paging read operations.

WRITES is the number of paging write operations.

(per sec) is the number of FAULTS, CREATES, READS, and WRITES that occurred each second.

PREPAGE shows prepaging statistics. The memory manager often transfers extra pages (along with the pages it needs to transfer) in order to reduce the number of page faults.

READS/USED is the number of times an extra page is brought in from a swap file and the number of page faults avoided.

WRITES is the number of extra pages written along with required writes.

CLOCK indicates statistics about the memory manager clock-replacement algorithm.

CALLS is the number of times the algorithm was invoked to obtain a page frame.

CYCLES (per sec)	is the number of times the algorithm has cycled through all main-memory page frames and the rate of cycles for each second. This rate is the reciprocal of the minimum page lifetime--the time it takes an unreferenced page to be removed from memory. For example, if CYCLES/SEC = .05, the minimum page lifetime is 20 seconds. In a balanced system, page lifetime should be in excess of 50 seconds; if CYCLES/SEC is greater than .02, then excessive paging is likely.
SCANS/CALL	is the mean number of page frames examined to find a replaceable page.
FAILURES	is the number of times the algorithm failed to find a replaceable page.

Produced by the MESSAGES Option

BUS SENDS	is the number of SEND instructions executed.
UNSEQUENCED	is the number of one-packet messages such as "I'm alive" and message acknowledgments.
CONTROL	is the number of message system control packets (that is, LCBs).
DATA	is the number of variable-length data messages.
MSG LEN	is the breakdown of data messages by length.
CNT	is the number of messages of specified length.

Produced by the INTERRUPTS Option

INTRPTS	counts the number of interrupt conditions that occurred. Interrupt conditions are further explained in the System Description Manual.
DISP	is the number of dispatcher interrupts.
BUS	is the number of bus receive interrupts.
HIIO	is the number of high priority I/O interrupts.
IIO	is the number of I/O interrupts.
TIME	is the number of time list interrupts.
FAULT	is the number of page fault interrupts.
SCHANL	is the number of special channel error interrupts.
CME	is the number of correctable memory error interrupts.
UCME	is the number of uncorrectable memory error interrupts.
MAB	is the number of memory access breakpoint interrupts.
BKPT	is the number of instruction breakpoint interrupts.
OSP	is the number of Operations and Service Processor interrupts.
PFAIL	is the number of power-fail interrupts.
PON	is the number of power-on interrupts.
TRAPS	is the number of traps (instruction failure, overflow, and stack overflow).

A study of the SYSGEN sections of the System Management Manual is recommended to familiarize yourself with the entities involved (such as TLEs, PCBs, and so on).

Example of PEEK Report

The following two-part example shows all the statistics for CPU 4, first for the elapsed time since the CPU was last loaded and then (by means of DYNAMIC) for the 10-second interval following. Since no OUT <list-file> is specified in the command, the output report defaults to the home terminal.

```
:PEEK / CPU 4 / 2, 10, ALL, DYNAMIC
```

```
9 OCT 1984 , 11:14___ELAPSD   13:05:17___CPU  4(TXP)    ___SAMP  1/2,DELAY  10

TIME:      PROCESSBUSY TIME          INTERRUPT TIME          IDLE TIME
           0:11:06.333   1.41%       0:06:29.560   0.82%     12:47:41.684  97.75%

           MAXIMUM USED  CURRENT USAGE  # CONFIGURED  # OF FAILURES
TLE               8             5            768             0
PCB             170           165            256             0
LCB              48             0            319             0
DCT             450           442           2708             0
PTLE              0             0              0             0

           MAX.SIZE  CUR.SIZE  CNF.SIZE  MAX.USED  CUR.USED  MAX.FRAG  CUR.FRAG
SYSPOOL         216        68     14274       216        52         2         1
MAPPOOL       19244     18414    260328     19244     16926         7         3

POOLGETS: 208,338     PUTS: 206,299     LOCKS: 216,435     UNLOCKS: 214,441

PAGES: PHYSCL  SWAPBL  FREE    LOCKED          FAULTS  CREATES    READS  WRITES
         2048    1875   769   248/1631          5828     2554     3273     829
                                          (per sec) 0.12     0.05     0.06    0.01
PREPAGE:READS/ USED  WRITES    CLOCK:CALLS  CYCLES(per sec) SCANS/CALL  FAILS
          2745/  1916    395          9911    12.22( 0.000 )     2.31      0

BUS SENDS: UNSEQUENCED: 1,222,501     CONTROL: 523,957      DATA: 153,323
MSG LEN: <=32b     <=128b     <=512b      <=1kb      <=2kb      <=4kb      >4kb
CNT:      74,740     42,250     33,474        140      1,953        631       135

INTRPTS: DISP         BUS          HIIO          IIO          TIME         FAULT
    1,540,252   1,918,292             0            0       253,904         7,744
   SCHANL    CME    UCME     MAB    BKPT     OSP   PFAIL     PON   TRAPS
        0      0       0       0       0       0       0       0     188
```

```
9 OCT 1984 , 11:14___ELAPSD    0:00:10___CPU  4(TXP)    ___SAMP  2/2,DELAY  10

TIME:      PROCESSBUSY TIME          INTERRUPT TIME          IDLE TIME
           0:00:00.090   0.83%       0:00:00.074   0.69%     0:00:10.596  98.46%

           MAXIMUM USED  CURRENT USAGE  # CONFIGURED  # OF FAILURES
TLE               8             5            768             0
PCB             170           165            256             0
LCB              48             0            319             0
DCT             450           442           2708             0
PTLE              0             0              0             0

            MAX.SIZE  CUR.SIZE  CNF.SIZE  MAX.USED  CUR.USED  MAX.FRAG  CUR.FRAG
SYSPOOL          216        68     14274       216        52         2         1
MAPPOOL        19244     18414    260328     19244     16926         7         3

POOLGETS: 12          PUTS: 12          LOCKS: 12          UNLOCKS: 12

PAGES: PHYSCL  SWAPBL  FREE    LOCKED          FAULTS  CREATES    READS  WRITES
         2048    1875   769   248/1631             1        0        1       0
                                          (per sec) 0.09     0.00     0.09    0.00
PREPAGE:READS/ USED  WRITES    CLOCK:CALLS  CYCLES(per sec) SCANS/CALL  FAILS
            1/     1       0             2     0.00( 0.000 )     0.00      0

BUS SENDS: UNSEQUENCED: 255          CONTROL: 93          DATA: 6
MSG LEN: <=32b     <=128b     <=512b     <=1kb     <=2kb     <=4kb     >4kb
CNT:           0          2          4         0         0         0        0

INTRPTS: DISP         BUS         HIIO          IIO         TIME        FAULT
           289         388            0            0           58            2
   SCHANL    CME    UCME     MAB    BKPT     OSP   PFAIL     PON   TRAPS
        0      0       0       0       0       0       0       0       0
```

In the first PEEK sample, you can see that CPU 4 is a NonStop TXP processor that has not been very busy overall (idle time is 90 percent), but the most recent activity in the last 10 seconds was quite high (idle time was only 14 percent).

The TLE and PCB statistics can give you an idea about how many TLEs and PCBS you should specify the next time you run SYSGEN; if the number configured approaches the maximum used, then problems can develop.

As described in the prior section on page locking, the cycles for each second (rounded to 0.000) is well below the 0.020 high paging-rate threshold. This is another confirmation that the physical memory in CPU 4 is adequate for processes running in this CPU.

The Messages and Interrupts portions of the PEEK report give you an idea of how busy the CPU is and can help in balancing the load across other CPUs. For example, if you find an excessive number of I/O interrupts for a CPU, you can use XRAY to determine the process causing them and move that process to another CPU.

SPOOL Command

The SPOOL command is used by system operators to run the spooler supervisor as part of cold starting the spooler. Refer to the System Operator's Guide for a description of this process.

```
SPOOL / IN <control-file-name> , [ OUT <log-file> , ]
        NAME $<supervisor-process-name>
        [ , PRI <execution-priority> ]
        [ , CPU <primary-cpu> ] [ / <backup-cpu> ]
        [ , <cold-start-parameters> ]
```

<control-file-name>

contains the names and attributes of the processes, locations, and devices that make up the system. It has the same form as a disc file name except that the file name consists of at most seven letters or digits.

<log-file>

is the name of the device, process, or disc file where the spooler logs messages.

If omitted or if for some reason the supervisor cannot communicate with the specified log file, messages are logged on the system console ($0).

If OUT is present but <log-file> is omitted, $0 is assumed to be the log file.

<supervisor-process-name>

is the process name of the supervisor. The name "$SPLS" is recommended, but not mandatory.

→

<execution-priority>

is the execution priority of the supervisor. The range is 1 to 199. Specifying a value greater than 199 causes the supervisor to run at priority 199.

The default is one less than the priority of the command interpreter from which the supervisor is run.

<primary-cpu>

is the number of the processor module (0 through 15) in which the supervisor's primary process runs.

The default is the same processor module as where the command interpreter is located.

<backup-cpu>

is the number of the processor module (0 through 15) in which the supervisor's backup is to run.

If omitted, the supervisor runs without a backup.

<cold-start-parameters>

are the following parameters. All of these are required when the the spooler is cold started. None of them can be included in a warm start. They cannot be changed without cold starting the spooler again.

<num-of-jobs>

is the maximum number of jobs that the spooler will handle at any one time. The range is 1 through 4095. There is no default.

→

<num-of-locations>

is the maximum number of destinations and groups the routing structure will contain. For example, the number of locations for a routing structure containing only the locations #DEFAULT.DEFAULT and #MYGRP.YRDEST must be at least four: two locations plus two groups. The range is 2 through 4095. There is no default.

<num-of-devices>

is the maximum number of devices that can be known to the spooler at any one time. The range is 1 through 255. There is no default.

<num-of-collectors>

is the maximum number of collectors that will ever be declared for this spooler. The range is 1 through 15. There is no default.

<num-of-print-processes>

is the maximum number of print processes that will ever be declared for this spooler. The range is 1 through 15. There is no default.

Considerations

- The supervisor forms the names of its control files by appending digits from "0" through "9" to the specified <control-file-name>.
- When cold starting a spooler, make sure that none of the <control-file-name> files exist, because the supervisor attempts to create them. If a file already exists, the supervisor will fail and terminate. The spooler may not use all of these files, but you should keep them reserved for use by the spooler.

- You must specify a $<supervisor-process-name>, because the spooler will not run an unnamed process. Tandem recommends that you use $SPLS as the name for the spool supervisor. Running SPOOLCOM without specifying a particular supervisor causes SPOOLCOM to communicate with a process named $SPLS.

- The process name assigned to a supervisor on a warm start need not be the same as the process name assigned to the supervisor the last time the spooler was made active. It is the control file name rather than the supervisor process name that uniquely identifies a particular spooler.

Example:

To create the spooler supervisor process, specifying a maximum of 4095 jobs and locations, and no more than 255 devices, 10 collectors, and 10 print processes, enter:

```
:SPOOL / IN $MKT.SPL.SPL, OUT $0, NAME $SPLS, NOWAIT, &
:PRI 147, CPU 0/1, 4095, 4095, 255, 255, 10, 10
```

INDEX

READER COMMENT CARD

Tandem welcomes your comments on the quality and usefulness of its software documentation. Does this manual serve your needs? If not, how could we improve it? Your comments will be forwarded to the writer for review and action, as appropriate.

If your answer to any of the questions below is "no," please supply detailed information, including page numbers, under Comments. Use additional sheets if necessary.

- ▶ Is this manual technically accurate? Yes ☐ No ☐
- ▶ Is information missing? Yes ☐ No ☐
- ▶ Are the organization and content clear? Yes ☐ No ☐
- ▶ Are the format and packaging convenient? Yes ☐ No ☐

Comments

Name Date

Company

Address

City/State Zip

GUARDIAN™ Operating System Utilities Reference Manual

NonStop™ Systems

82403 A00

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA, U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

Tandem Computers Incorporated
Attn: Manager—Software Publications
Location 01, Department 6350
19333 Vallco Parkway
Cupertino CA 95014-9990

TAPE TAPE

Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, CA 95014-2599