Prime®

PRIMOS® Commands Reference Guide

Revision 22.0

DOC3108-7LA

PRIMOS® Commands Reference Guide

Seventh Edition

Beth Byers
William Carbonneau
Daniel Laukaitis

Prime Computer, Inc.
Prime Park
Natick, Massachusetts 01760

Copyright Information

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

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

Copyright © 1988 by Prime Computer, Inc., Prime Park, Natick, Massachusetts 01760

PRIME, PR1ME, PRIMOS, and the PRIME logo are registered trademarks of Prime Computer, Inc. DISCOVER, EDMS, FM+, INFO/BASIC, INFORM, Prime INFORMATION, Prime INFORMATION CONNECTION, MDL, MIDAS, MIDASPLUS, PRIME MEDUSA, PERFORM, PERFORMER, PRIME/SNA, PRIME TIMER, PRIMECALC, PRIMELINK, PRIMENET, PRIMEWAY, PRIMEWORD, PRIMIX, PRISAM, PRODUCER, Prime INFORMATION/pc, PST 100, PT25, PT45, PT65, PT200, PW153, PW200, PW250, RINGNET, SIMPLE, 50 Series, 400, 750, 850, 2250, 2350, 2450, 2550, 2650, 2655, 2755, 6350, 6550, 9650, 9655, 9750, 9755, 9950, 9955, and 9955II are trademarks of Prime Computer, Inc.

UNIX is a registered trademark of AT&T.

Printing History

First Edition (FDRR3108-101) January 1979 for Revision 16.3
Second Edition (FDR3108-101) May 1980 for Revision 17.2
Third Edition (FDR3108-101) January 1981 for Revision 18.1
Fourth Edition (FDR3108-101) July 1982 for Revision 19.0
Fifth Edition (DOC3108-5LA) March 1985 for Revision 19.4
Sixth Edition (DOC3108-6LA) July 1987 for Revision 21.0
Seventh Edition (DOC3108-7LA) October 1988 for Revision 22.0

Credits

Editorial: Barbara Fowlkes, Kathe Rhoades
Project Support: Frank Calvillo, Joan Karp, Donald Markuson, John Seybold
Design: Leo Maldonado
Production: Judy Gordon
Graphics: Julie Cyphers, Anne Marie Fantasia
Document Preparation: Mary Mixon, Katherine Normington
Composition: Julie Cyphers, Anne Marie Fantasia, Sharon Temple

This guide documents the software operation of the Prime Computer and its supporting systems and utilities as implemented at Master Disk Revision Level 22.0 (Rev. 22.0).

How to Order Technical Documents

To order copies of documents, or to obtain a catalog and price list:

United States Customers

Call Prime Telemarketing, toll free, at 1-800-343-2533, Monday through Friday, 8:30 a.m. to 5:00 p.m. (EST).

International

Contact your local Prime subsidiary or distributor.

Customer Support

Prime provides the following toll-free numbers for customers in the United States needing service:

1-800-322-2838 (within Massachusetts)
1-800-541-8888 (within Alaska and Hawaii)
1-800-343-2320 (within other states)

For other locations, contact your Prime representative.

Surveys and Correspondence

Please comment on this manual using the Reader Response Form provided in the back of this book. Address any additional comments on this or other Prime documents to:

Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, MA 01701

The PRIMOS operating system, Revision 22.0, is a proprietary software product of Prime Computer, Inc., and is distributed by Prime exclusively under a licensing agreement protecting its copyrighted and trade secret features.

The term PRIMOS encompasses not only the basic operating system, but also a collection of software utilities, tools, and modules that are integrated with and support the operating system. These additional software items include the following:

$$.RUN
ADMIN_LOG.RUN
AVAIL.SAVE
BATCH.RUN
BATGEN.RUN
BIND.RUN
BOOT_ATTACH.CPL
BOOT_CREATE.CPL
BOOT_CREATE.MTTYPE.SAVE
BOOT_IMPCODE.CPL
BOOT_SAVE.CPL
BOOT_TREE.CPL
CAB.RUN
CMPF.SAVE
COMM_CONTROLLER.RUN
CONCAT.SAVE
CONFIG.CPL
CONFIG_DSM.RUN
CONFIG_UM.RUN
COPY.RUN
COPY_DISK.SAVE
CPMPC.SAVE
CRMPC.SAVE
DELETE.RUN
DISPLAY_LOG.RUN
DISTRIBUTE_DSM.RUN
ECL.CPL
ED.SAVE
EDB.SAVE
EDIT_CMD_LINE.RUN
EDIT_PROFILE.SAVE
ESR.RUN
EXPAND_SEARCH_RULES.RUN
FILMEM.RUN
FILVER.SAVE
FIX_DISK.SAVE
FUTIL
HELP.RUN
IFB.RUN
JOB.RUN
KLMD.RUN
KLMF.RUN
KLMT.RUN
L.CPL
LAB.RUN
LABEL.RUN
LATE.SAVE
LCB.RUN
LD.RUN
LEM.RUN
LI.CPL
LIS.CPL
LIST.CPL
LISTF.CPL
LISTI.CPL
LISTIN.CPL
LISTING.CPL
LIST_ASSIGNED_DEVICES.RUN
LIST_ASYNC.RUN
LIST_COMM_CONTROLLERS.RUN
LIST_CONFIG.RUN
LIST_CONTIGUOUS_BLOCKS.RUN
LIST_DISKS.RUN
LIST_LAN_NODES.RUN
LIST_MEMORY.RUN
LIST_PRIMENET_LINKS.RUN
LIST_PRIMENET_NODES.RUN
LIST_PRIMENET_PORTS.RUN
LIST_PROCESS.RUN
LIST_PRIMENET_PORTS.RUN
LIST_SERVER_NAMES.RUN
LIST_SESSIONS.RUN
LIST_SYNC.RUN
LIST_UNITS.RUN
LIST_VCS.RUN
LOAD.SAVE
MAGNET.RUN
MAGRST.SAVE
MAGSAV.SAVE
MAKE.SAVE
MRGF.SAVE
NSED
PHYRST.SAVE
PHYSAV.SAVE
PMA.SAVE
PRMPC.SAVE
PRO.CPL
PROP.RUN
PROT.CPL
PROTE.CPL
PROTEC.CPL
PROTECT.RUN
PRSER.SAVE
PRVER.SAVE
RESUS.RUN
REVERT_PASSWORD.RUN
RUNOFF.RUN
RWLOCK.RUN
SEG.SAVE
SET_ASYNC.RUN
SET_DELETE.RUN
SIZE.RUN
SLIST.SAVE
SORT.RUN
SPOOL.RUN
START_DSM.RUN
START_LSR.RUN
STATUS_DSM.RUN
STOP_DSM.RUN
STOP_LSR.RUN
TERM.RUN
TRAMLC.SAVE
USAGE.SAVE
UX_TAPE.RUN

Contents

1	**Introduction**	1-1
	Overview	1-1
	Names of File System Objects	1-2
	PRIMOS Command-line Format	1-5
	Command Format Conventions	1-6
	Conventions in Examples	1-7
	Advanced Command-line Functionality	1-8
	Summary of PRIMOS Commands	1-9
	Summary of User Commands by Function	1-11
	Summary of Operator Commands	1-24
	Summary of Command Functions	1-25
	Summary of Changes in Commands	1-26
2	**Dictionary of PRIMOS Commands**	2-1
3	**Dictionary of Command Functions**	3-1
	Introduction	3-1
	Command Functions	3-2
4	**Command-line Features**	4-1
	Introduction	4-1
	Global Variables	4-2
	Multiple Commands	4-2
	Iteration	4-3
	Wildcarding	4-5
	Treewalking	4-12
	Name Generation	4-19
	Combining Command-line Features	4-22
	Syntax Suppression	4-23

5 Command-line Processing 5-1
Introduction 5-1
Command-Line Processing Order 5-1
Example of Command-line Processing 5-7

6 Command-line Editing 6-1
Introduction 6-1
Automatic Pathname Completion 6-3
ECL Key Bindings 6-8
Commands Useful for ECL Key Bindings 6-11
Programmable Function Commands 6-12
Sample Bindings File 6-15
Bindings for Terminals 6-16
Compiling, Saving, and Restoring the Bindings File 6-23
Display Manager 6-27
ECL Commands Reference 6-31

Appendices

A RVEC Parameters A-1
Meaning of RVEC Parameters A-1
Meaning of the Keys Parameter A-2

B DUMP_STACK Format B-1
DUMP_STACK Command B-1
Format of Stack Frames B-1

C Old Commands C-1
Introduction C-1
Passwords C-2
Logical Disk C-3
Home Key Values C-3

D ECS & EBCDIC Character Sets D-1
Specifying Prime ECS Characters D-1
Special Meanings of Prime ECS Characters D-2
Prime Extended Character Set Table D-2

E Commands By Function E-1
Cursor Movement E-1
Insertion E-2
Deletion E-2
Deletion Recall E-3
Regions E-3
Repetition E-4
Twiddle E-4
Submit and Abort E-4
Case Conversion E-5
Pathname and Abbreviation Expansion E-5
History and Refresh E-6
Keyboard Macros E-6
Binding and PFkey Support E-7
Miscellaneous E-7

Index X-1

1 Introduction

Overview

The *PRIMOS Commands Reference Guide* is intended for the user who is working on a Prime® computer and who needs detailed information on a particular PRIMOS® command. This guide is not intended as an introduction to PRIMOS, the Prime operating system. Introductory material to PRIMOS is supplied in the *PRIMOS User's Guide*.

This guide provides the following types of information:

- Detailed information on PRIMOS commands available to the user
- A brief description of other PRIMOS commands (such as those available to operators or those that invoke separately priced products) and references to detailed information about those commands

The book is divided into the following seven sections:

- Chapter 1, Introduction, provides a brief review of the PRIMOS command line, explains the book's conventions for displaying command-line formats and examples, and summarizes all PRIMOS operator and user commands.
- Chapter 2, Dictionary of PRIMOS Commands, provides an alphabetical listing of commands, each entry containing either detailed instructions for the use of the command or a reference to the book in which a detailed description exists.
- Chapter 3, Dictionary of Command Functions, describes the function calls available for use in the command line.
- Chapter 4, Command-line Features, describes the PRIMOS command-line features.
- Chapter 5, Command-line Processing, describes the order in which command-line processing occurs.
- Chapter 6, Command-line Editing, describes the PRIMOS command-line editor (ECL).
- Appendices A through E provide further details on miscellaneous matters.

Names of File System Objects

The following sections define terms that are used in this guide to refer to the names of file system objects. A **file system object** is a file, directory (either top-level directory or subdirectory), segment directory, or access category.

Diskname

A **disk** (also called a **partition**) is a part of or all of a multihead disk pack. Each disk must have a unique name because PRIMOS treats each disk as a separate logical device. Disknames are composed of six or fewer characters.

A **diskname** is the first element in an absolute pathname, and is the only element in the pathname that is enclosed in left and right angle-brackets (< >). In the following pathname, FOREST is the diskname:

```
<FOREST>OAK>BRANCH>LEAF
```

Synonyms sometimes used for diskname are partition name, device name, pack name, and volume name.

To list the disks connected to your system, use the STATUS DISKS command. The disknames are in the leftmost column of the command's output.

Pathnames

A **pathname** specifies a path to a file system object. The path begins at the diskname, descends through one or more directories, and ends at the entryname.

The pathname contains two or more names, each name separated from the previous one by a right angle-bracket (>). The right angle-bracket indicates that there is a lower level in the path. The first name in the pathname is the diskname (although the name may be implied), and the last name is always an entryname. Names between the diskname and the entryname are directories.

There are three types of pathnames: absolute pathnames, full pathnames, and relative pathnames. The difference among them is the completeness of the path. Each type is described below.

Absolute Pathname: An **absolute pathname**, which is the most complete type of pathname, begins with the diskname. The following is an example of an absolute pathname:

```
<FOREST>OAK>BRANCH>LEAF
```

FOREST is the diskname, OAK is a top-level directory, BRANCH is a subdirectory of OAK, and LEAF is the entryname of a file system object.

Full Pathname: A **full pathname** begins with a top-level directory. The diskname is omitted and is assumed to be the disk on which PRIMOS finds the top-level directory. (See the ATTACH command in Chapter 2 of this guide for the order in which PRIMOS searches disks for top-level directories.) A full pathname may also contain one or more subdirectories.

Here is an example of a full pathname:

```
OAK>BRANCH>LEAF
```

OAK is a top-level directory, BRANCH is a subdirectory of OAK, and LEAF is the entryname. The diskname (in this case, FOREST) is not part of the full pathname but is assumed to be the disk with the lowest logical device number that contains a top-level directory named OAK.

Relative Pathname: A **relative pathname** begins with the current directory, which is represented by an asterisk (*). The asterisk means that the part of the pathname down to and including the current directory is implied. For example, if <FOREST>OAK>BRANCH>LEAF were the absolute pathname of the object LEAF, then the following would be LEAF's relative pathname if OAK were the current directory:

```
*>BRANCH>LEAF
```

If BRANCH were the current directory, then the relative pathname of LEAF would be

```
*>LEAF
```

You can also use an asterisk to stand for the current diskname. For example, if two disks had top-level directories named OAK and you wanted to attach to the OAK directory on your current disk, you would use the following relative pathname to make certain you arrived at the correct directory:

```
OK, ATTACH <*>OAK
```

Entryname

An **entryname** is the final element of a pathname. (That is, the entryname part of a pathname consists of everything to the right of the last right angle-bracket). For example, LEAF is the entryname in the pathname <FOREST>OAK>BRANCH>LEAF.

Entryname is the generic term for the name of any file system object. This term is used because every object is an entry in a directory. When you use the LD command without arguments, it displays the entries in that directory.

The following terms refer to specific types of file system objects:

- **Categoryname** is the entryname of an access category.
- **Directoryname** is the entryname of a directory.
- **Filename** (also called a simple filename) is the entryname of a file.
- **Objectname** is a synonym for entryname.

An entryname can be made up of two or more components if each component is separated with a period. Such an entryname is composed of a basename and one or more suffixes.

Basename and Suffix: When an entryname has two or more components, everything up to the last period is the basename. The last component is the suffix. For example, in the pathname LAB>PROGRAMS>PRIME_NUMBER.FTN, the entryname PRIME_NUMBER.FTN is made up of the basename PRIME_NUMBER and the suffix .FTN.

Suffixes are particularly useful for keeping track of different types of objects in a directory. Some suffixes are also recognized by Prime software. The following is a partial list of suffixes recognized by Prime software:

Suffix	*Type of Object*
ACAT	Access category
BASIC	BASIC/VM source file
BIN	Binary file created by a compiler
CBL	COBOL 74 source file
CDML	COBOL Data Manipulation Language preprocessor input file
CPL	CPL source file
DPTCFG	DPTCFG input file
EFASL	EMACS fastload file
EM	EMACS extension file
ERROR	Error file
F77	FORTRAN 77 source file
FBIN	FORMS binary file
FORM	FORMS file
FTN	FORTRAN IV source file
LIST	Listing file created by a compiler
PASCAL	Pascal source file
PL1	PL/I source file
PL1G	PL/I Subset G source file
PMA	PMA source file
RUN	EPF runfile created by the BIND linker

SAVE	Static R-mode runfile created by the LOAD loader
SEG	Segment directory (static V-mode runfile) created by SEG
VRPG	RPG II source file (used for the V-mode VRPG compiler)

For further information on suffix-naming conventions, see the *PRIMOS User's Guide*.

PRIMOS Command-line Format

The general format of the PRIMOS command line is

COMMAND [names] [–OPTION argument [... –OPTION argument]]

COMMAND specifies one of the PRIMOS commands in Chapter 2. The command must be the first word on the command line. **names** specifies the argument (or arguments) to the command. The command arguments are generally pathnames of directories or files, or an identifying name such as a user ID or a job name. **–OPTION** specifies a command option. (Some commands do not have options.)

For example, the format of the COPY command is

COPY source-pathname [target-pathname] [options]

In this example, **source-pathname** and **target-pathname** are both arguments to the command. **source-pathname** tells PRIMOS the name of the file to be copied. **target-pathname**, which is optional, provides a new pathname for the file. **options** lets you specify additional instructions concerning the copy procedure, such as deletion of the original file after the new copy is made.

The example below shows a COPY command using the –DELETE option:

```
COPY STATUS.Q2 SALES>QUARTER_RPT -DELETE
```

Using Pathnames and Entrynames as Arguments

When a command format calls for a pathname, you can usually use a full or relative pathname instead of an absolute pathname. If the file system object is in your current directory, you can simply use the entryname.

In the COPY command above, for example, STATUS.Q2 is an entryname (a filename in this case) because it is in the current directory. SALES>QUARTER_RPT is a full pathname because the copy is to be placed in another directory.

Length of Command Lines

Command lines can be a maximum of 160 characters long. Command lines that exceed that length are rejected by the command processor, with the following error message:

```
Command-line longer than 160 characters. (listen_)
ER!
```

Case of Command-line Words

Unless specified otherwise in the command description, you can enter the command line in either uppercase or lowercase because PRIMOS converts lowercase characters to uppercase.

Command Format Conventions

The conventions for PRIMOS command documentation in this book are as follows:

Convention	*Meaning*
▶	A solid, red triangle identifies a command or command function entry in Chapters 2 and 3.
▷	An empty, red triangle identifies a subcommand entry in Chapter 2. A subcommand is a command given within a program that was invoked by a PRIMOS command (for example, CONCAT or ED).
UPPERCASE	Capitalized words in boldface identify a command name (such as **COPY** or **CLOSE**) or a mandatory keyword (such as the **MT** keyword for the LABEL command). They must be entered literally. If a portion of the word is in red letters, the red letters indicate the system-defined abbreviation.
lowercase	Lowercase words in boldface identify the following: (1) variables that serve as arguments to commands or options and for which the user substitutes an appropriate numerical or text value; or (2) new terms that are subsequently defined.
–WORD	An uppercase word preceded by a hyphen indicates a command-line option (for example, –LIST in the SPOOL –LIST command format). The hyphen must be included and cannot be separated from the word by spaces. As with command names, if a portion of the option name is in red letters, the red letters indicate the system-defined abbreviation.
options	The word **options** indicates that one or more options can be specified for a command and that a list of options for the particular command follows. Command-line options usually are preceded by a hyphen. (See –WORD above.)

Italics	Words in italics identify the following: (1) titles of other Prime documents to which the user is referred; or (2) the term *Operator Command*, which identifies a PRIMOS command as an operator command; (3) comment lines in examples.
Braces { }	Braces indicate a choice of arguments and/or options. Choose one and only one of these items.
Square Brackets [ ]	Square brackets shown in black enclose a list of one or more optional items; choose none, one, or several of these items. Square brackets shown in red indicate a function call; these must be entered literally.
Braces Within Brackets [{ }]	Braces within brackets enclose a list of items. Choose either none or only one of these items; do not choose more than one.
Vertical Bars ||	Vertical bars enclose a list of items. Choose one or more of these items.
Subscript 200_8	A subscript after a number indicates that the number is not in base 10. For example, the subscript 8 is used for octal numbers.
Ellipsis	An ellipsis indicates that the preceding argument or option may be repeated.
Angle Brackets < >	Angle brackets separate the elements of the pathname of a file system object, for example, <FOREST>BEECH>BRANCH5>TWIG2.
Quotation Marks " "	Text within double quotation marks indicates a message that appears at the terminal.
Parentheses ( )	Parentheses are used literally in an iteration list. For example, the command DELETE (BROWN GREEN) deletes two file system objects named BROWN and GREEN.
Spaces	Command words, arguments, and options are separated in command lines by one or more spaces. To contain a literal space, an argument must be enclosed in single quotation marks.

Conventions in Examples

In all examples, the user's input is indicated by red text, while the system's output is in black text. Explanatory notes are in italics. For example,

```
OK, attach *>examples        /*attach to subdirectory
OK, ed seginfo          /*invoke the editor
EDIT
```

User input in examples usually may be either in lowercase or uppercase letters. The rare exceptions are specified in the commands where they occur.

Advanced Command-line Functionality

The PRIMOS command line supports the following features:

- User-defined abbreviations
- Command-line syntax suppression
- Multiple commands on one line
- User-defined global variables
- PRIMOS command functions
- Command iteration
- Wildcard names
- Treewalk pathnames
- Name generation patterns

User-defined abbreviations are explained in the discussion of the ABBREV command in Chapter 2. Command functions are described in Chapter 3. The other features are discussed in Chapters 4 and 5.

The command processor recognizes these features by looking for special characters on the command line. These special characters, in the order they are searched for, are given in Table 1-1. User-defined abbreviations, which are processed first, use no special characters.

TABLE 1-1
Special Characters on the Command Line

Special Character	*Feature*
~	Syntax suppression: disables the effect of all following special characters. Must be in the first position of the line.
;	Command separator: allows several commands to be given at once.
% %	Global variables: the variable must be enclosed within percent signs.
[ ]	Command functions.
( )	Iteration.
@ @@ + ^	Treewalking: the wild characters must be in an intermediate position of the pathname. Use only one negation character (^) on the command line.
@ @@ + ^	Wildcard name: the wild characters must be in the final position of the pathname. Use only one negation character (^) on the command line.
= == ^= ^== +	Name generation: specifies a generation pattern.

In general, when abbreviations, global variables, and command functions appear on the command line, the command-line processor substitutes the value of the item for the item itself. This creates a one-to-one substitution.

However, when an iteration list, a wildcard name, or a treewalk name is found, the command processor creates one command for each item it finds or matches on the list. In iteration lists, you supply each item explicitly. In wildcard and treewalk names, you set the pattern, and the command processor searches the specified directory or directories for all file system objects that match that pattern. Thus, these features can be thought of as creating one-to-many matches.

You can use a name generation pattern to create a matching name either for a single filename or for a number of filenames that result from a wildcard or treewalking pattern.

Note

Not all commands support all the features listed above. The general rule is that if a feature is not useful in connection with a particular command, the command does not support it.

Summary of PRIMOS Commands

There are two kinds of PRIMOS commands: user commands and operator commands. A summary of each type of command is given in the two following sections.

Each PRIMOS command is either an internal command or an external command. **Internal commands** are part of PRIMOS itself. **External commands** are actually programs that are stored in a special top-level directory named CMDNC0. Some external commands invoke separately priced software products that may be on your system. Moreover, System Administrators may add or remove external commands to meet the needs of their particular systems. For these reasons, not every system recognizes all the external commands listed in this book.

To find out what external commands are available on your system, use the ATTACH command to attach to CMDNC0 and then use the LD command to list its files. (This procedure works only if the System Administrator allows users to look at CMDNC0.) An example of this procedure is shown below:

```
OK, A CMDNC0
OK, LD

<SYSONE>CMDNC0 (LU access)
2853 records in this directory, 2853 total records out of quota of 0.

115 Files.

$$.RUN                  ADMIN_LOG.RUN           AVAIL.SAVE              BATCH.RUN
BATGEN.RUN              BIND.RUN                BOOT_ATTACH.CPL         BOOT_CREATE.CPL
BOOT_CREATE.MTTYPE.SAVE                         BOOT_IMPCODE.CPL        BOOT_SAVE.CPL
BOOT_TREE.CPL           CAB.RUN                 CMPF.SAVE               COMM_CONTROLLER.RUN
CONCAT.SAVE             CONFIG.CPL              CONFIG_DSM.RUN          CONFIG_UM.RUN
COPY.RUN                COPY_DISK.SAVE          CPMPC.SAVE              CRMPC.SAVE
DELETE.RUN              DISPLAY_LOG.RUN         DISTRIBUTE_DSM.RUN      ECL.CPL
ED.SAVE                 EDB.SAVE                EDIT_CMD_LINE.RUN       EDIT_PROFILE.SAVE
ESR.RUN                 EXPAND_SEARCH_RULES.RUN                         FILMEM.RUN
FILVER.SAVE             FIX_DISK.SAVE           FUTIL                   HELP.RUN
IFB.RUN                 JOB.RUN                 KLMD.RUN                KLMF.RUN
KLMT.RUN                L.CPL                   LAB.RUN                 LABEL.RUN
LATE.SAVE               LCB.RUN                 LD.RUN                  LEM.RUN
LI.CPL                  LIS.CPL                 LIST.CPL                LISTF.CPL
LISTI.CPL               LISTIN.CPL              LISTING.CPL             LIST_ASSIGNED_DEVICES.RUN
LIST_ASYNC.RUN          LIST_COMM_CONTROLLERS.RUN                       LIST_CONFIG.RUN
LIST_CONTIGUOUS_BLOCKS.RUN                      LIST_DISKS.RUN          LIST_LAN_NODES.RUN
LIST_MEMORY.RUN         LIST_PRIMENET_LINKS.RUN                         LIST_PRIMENET_NODES.RUN
LIST_PRIMENET_PORTS.RUN                         LIST_PROCESS.RUN        LIST_SEMAPHORES.RUN
LIST_SERVER_NAMES.RUN                           LIST_SESSIONS.RUN       LIST_SYNC.RUN
LIST_UNITS.RUN          LIST_VCS.RUN            LOAD.SAVE               MAGNET.RUN
MAGRST.SAVE             MAGSAV.SAVE             MAKE.SAVE               MODULA.RUN
MRGF.SAVE               NSED                    PHYRST.SAVE             PHYSAV.SAVE
PMA.SAVE                PRMPC.SAVE              PRO.CPL                 PROP.RUN
PROT.CPL                PROTE.CPL               PROTEC.CPL              PROTECT.RUN
PRSER.SAVE              PRVER.SAVE              RESUS.RUN               REVERT_PASSWORD.RUN
RUNOFF.RUN              RWLOCK.RUN              SEG.SAVE                SET_ASYNC.RUN
SET_DELETE.RUN          SIZE.RUN                SLIST.SAVE              SORT.RUN
SPOOL.RUN               START_DSM.RUN           START_LSR.RUN           STATUS_DSM.RUN
STOP_DSM.RUN            STOP_LSR.RUN            TERM.RUN                TRAMLC.SAVE
USAGE.SAVE              UX_TAPE.RUN

OK,
```

CMDNC0 files have various suffixes. Commands using the .RUN suffix behave like internal commands.

Summary of User Commands by Function

The following list summarizes all user commands commonly recognized by PRIMOS. The commands are grouped by function so that you can find which command is necessary for a particular situation. Some commands are listed under more than one category because they work on different types of file system objects. All commands in this summary are discussed in more detail in Chapter 2.

Establishing Your Identity and Accessing the System

ADD_REMOTE_ID	Specifies a user ID to be used for remote file access. (Internal)
ATTACH	Moves your location from one directory to another. (Internal)
CHANGE_PASSWORD	Replaces your login password with another login password of your choice. (Internal)
EDIT_PROFILE	Allows System and Project Administrators to add, change, and delete information about user profiles and the projects to which users belong. (External)
LIST_GROUP	Lists the access groups to which you belong. (Internal)
LIST_REMOTE_ID	Displays your current set of remote IDs. (Internal)
LOGIN	Begins a terminal session by identifying the user to the system and establishing the initial contact between system and user. (Internal)
LOGOUT	Ends a terminal session, including closing all files and releasing the PRIMOS process to another user. (Internal)
ORIGIN	Moves your location back to your origin directory (the directory to which you were attached when you logged in). (Internal)
REMOVE_REMOTE_ID	Cancels one of your remote IDs. (Internal)

Managing Directories

CNAME	Changes the name of a directory. (Internal)
COPY	Copies a directory. (External)
CREATE	Creates and names a new directory. (Internal)
DELETE	Deletes a directory or segment directory. (External)
LD	Lists the contents of a directory. (External)
LIST_QUOTA	Lists current disk quota and storage information for a specified directory. (Internal)

SET_DELETE	Sets the delete switch on a directory so that it cannot be deleted accidentally. (External)
SET_QUOTA	Defines the maximum number of records a directory and its subtree may use. (Internal)
SIZE	Displays the number of entries in a directory. (External)

Protecting Directories

With Access Control Lists:

EDIT_ACCESS	Changes protection on an ACL-protected directory by modifying its ACL. (Internal)
LIST_ACCESS	Displays access protection for a specified directory. (Internal)
LIST_PRIORITY_ACCESS	Displays the priority access in effect on a specified disk partition. (Internal)
SET_ACCESS	Sets access control protection on a directory. If the directory is a password-protected directory, converts it to an ACL-protected directory. (Internal)

With Passwords:

PASSWD	Replaces existing passwords on the current directory with new passwords. (Internal)
PROTECT	Defines rights of others to access the user's directory. (External)
REVERT_PASSWORD	Converts the current ACL-protected directory to a password-protected directory. (External)

Managing Files

General Management of Normal Files:

CNAME	Changes the name of a file. (Internal)
COPY	Makes a copy of a file. (External)
DELETE	Deletes a file. (External)
PLOT	Plots a metafile or device-specific plot file. (External)
SIZE	Displays the size of a file. (External)
SLIST	Displays the contents of a file on your terminal. (External)
SPOOL	Prints a file on the system printer or plotter. (External)

General Management of ROAM Files:

CN_RBF	Changes the name of an active ROAM master or slave file. (External)
COPY_RBF	Makes a copy of a ROAM file. (External)
DELETE_RBF	Deletes an active or inactive ROAM file. (External)
LIST_RBF	Lists the attributes of a ROAM master or slave file. (External)
REST_RBF	Restores an inactive ROAM file. (External)
SAVE_RBF	Archives a ROAM file to disk. (External)
SET_RBF	Defines the recovery and concurrency attributes of a ROAM file. (External)

Opening and Closing Files:

BINARY	Opens a file for writing on PRIMOS File Unit 3, usually as a binary output file for use by a compiler or assembler. (Internal)
CLOSE	Closes a file. (Internal)
COMINPUT	Opens or closes a command input file, usually on File Unit 6. (Internal)
COMOUTPUT	Opens or closes a command output file for recording interactive terminal input and output. (Internal)
INPUT	Opens a source file for reading on File Unit 1. (Internal)
LISTING	Opens a file for writing on File Unit 2. (Internal)
OPEN	Opens a file unit for reading, writing, or updating. (Internal)

Comparing and Modifying Files:

CMPF	Compares as many as five ASCII files and lists any discrepancies among them. (External)
CONCAT	Combines a number of ASCII files into one file; accepts formatting commands for spooling the file. (External)
FILVER	Compares two files and lists the differences between them (used primarily for runfiles). (External)
MRGF	Merges as many as five ASCII files, allowing the user to resolve conflicts among them. (External)
NUMBER	Numbers or renumbers statements in a BASIC program. (External)
SORT	Sorts up to 20 files into a single output file. (External)
UPCASE	Creates an uppercase-only file from a file containing both uppercase and lowercase characters. (External)

Protecting Files:

EDIT_ACCESS	Modifies the access control list protecting a file. (Internal)
LIST_ACCESS	Displays the contents of the access control list protecting a file. (Internal)
PROTECT	Sets user rights on a file in a password-protected directory. (External)
RWLOCK	Determines the number of readers and/or writers who can access a file at the same time. (External)
SET_ACCESS	Creates an access control list to protect a file. (Internal)
SET_DELETE	Sets the delete switch on a file so that it cannot be deleted accidentally. (External)

Editors and Text Handlers

ED	Invokes EDITOR, the Prime line-oriented editor. (External)
EDB	Invokes the Prime binary editor (used primarily for building and maintaining subroutine libraries). (External)
EMACS	Invokes EMACS, a character-oriented screen editor. (External)
RUNOFF	Invokes RUNOFF, the Prime text-formatting utility. (External)

Compilers, Translators, and Interpreters

BASIC	Invokes the older Prime BASIC interpreter. (External)
BASICV	Invokes the newer Prime virtual-memory BASIC subsystem (BASIC/VM). (External)
CC	Compiles a C program. (External)
CBL	Compiles a COBOL program, using the Prime COBOL 74 compiler. (External)
DBASIC	Invokes an interpretive BASIC with double-precision arithmetic capabilities. (External)
F77	Compiles a FORTRAN IV or FORTRAN 77 source program, using the FORTRAN 77 compiler. (External)
FTN	Compiles a FORTRAN IV program. (External)
PASCAL	Compiles a Pascal program. (External)
PL1	Compiles a PL/I program. (External)
PL1G	Compiles a PL/I, Subset G, program. (External)
PMA	Assembles a program written in the Prime Macro Assembler language (PMA). (External)

VRPG	Compiles an RPG II program, using the newer Prime V-mode RPG compiler. (External)

Linking and Executing Programs

BIND	Links V-mode and I-mode programs to create Executable Program Format (EPF) dynamic runfiles (.RUN files). (External)
DELSEG	Frees previously used segments. (Internal)
FILMEM	Fills memory locations 100 through 32K with zeroes (used before invoking LOAD). (External)
LOAD	Creates R-mode static runfiles. (External)
REENTER	Reenters a subsystem following a CONTROL-P or an error condition. (Internal)
REMOVE_EPF	Removes a mapped EPF from your address space. (Internal)
RESTOR	Restores a runfile from disk to memory. (Internal)
RESUME	Executes a CPL program or an R-mode program. (Internal)
SAVE	Saves (in a file) the contents of a specified location in memory. (Internal)
SEG	Creates and/or executes a V-mode or I-mode static runfile (.SEG files). (External)
START	Starts a program loaded by the RESTOR or the RESUME command, or restarts a program halted by a CONTROL-P or an error condition. (Internal)

Debuggers

DBG	Invokes the source level debugger, for debugging high-level language programs. (External)
IPSD	Invokes an extended version of the Prime Symbolic Debugger that supports S-mode, R-mode, V-mode, and I-mode programs. (External)
PSD **PSD20**	Invoke versions of the Prime Symbolic Debugger for debugging R-mode and S-mode assembler programs (or object code from high-level language programs). (External)
VPSD **VPSD16**	Invoke versions of the Prime Symbolic Debugger for debugging V-mode and I-mode assembler programs or object code. (External)

Job Processors

CPL	Executes a CPL program. (Internal)
COMINPUT	Executes a command input file. (Internal)
COMOUTPUT	Creates a file for recording interactive terminal input and output. (Internal)
JOB	Executes a CPL program or command input file as a Batch job. (External)
PHANTOM	Executes a CPL program or command input file as a phantom. (Internal)
$	Passes job information to the Batch monitor from inside CPL or command input files. (External)

Setting Your Terminal's Characteristics

DELAY	Defines a time function that delays the printing of a character after a carriage return (RETURN) has been output to a terminal. (Internal)
LATE	Forces the terminal to ignore commands until a specified time. (External)
TERM	Defines the terminal's erase and kill characters and duplex mode and enables (or disables) the operation of the CONTROL-P (or BREAK), CONTROL-S, and CONTROL-Q key sequences. (External)

Defining Your Command Environment

ABBREV	Defines abbreviations for PRIMOS commands and their arguments. (Internal)
CHAP	Changes your execution priority level. (Internal)
DEFINE_GVAR	Creates, activates, or deactivates a global variable file. (Internal)
DELETE_VAR	Deletes one or more variables from a global variable file. (Internal)
EXPAND_SEARCH_RULES	Provides the full pathname of a specified file system object or search list. (External)
INITIALIZE_COMMAND_ENVIRONMENT	Resets your command environment to the state it was in at your login. (Internal)
LIST_VAR	Lists variables from a global variable file. (Internal)
LON	Enables or disables phantom logout notification. (Internal)
RDY	Selects the prompt message you want displayed at your terminal. (Internal)

RELEASE_LEVEL	Frees space by discarding unwanted stack history. (Internal)
SET_SEARCH_RULES	Sets a search list to user-specified values. (Internal)
SET_VAR	Adds a variable to a global variable file. (Internal)

Displaying System Information

AVAIL	Displays the amount of disk space being used. (External)
BATGEN	Provides information about Batch queues. (External)
DATE	Displays the current date and time at your terminal. (Internal)
DPTXMTR	Provides information about the DPTX queues and events on the communication lines. (External)
HDXSTAT	Displays information about the current status of all lines and sites of a half-duplex (HDX) network configuration. (External)
HELP	Displays online information about PRIMOS commands. (External)
LCB	Provides information about available space on a partition. (External)
LEM	Lists the extent map of a CAM subfile. (External)
PROP	Provides information about system printers and/or plotters. (External)
STATUS	Displays general information, such as what users are logged in, what disks are running, what nodes on a network are available, and so on. (Internal)
USAGE	Meters system usage. (External)
USERS	Displays the number of users currently logged in. (Internal)

Information on Your Current Environment

AVAIL	Lists records used and available on your current disk. (External)
LIST_ACCESS	Lists user access on your current directory. (Internal)
LIST_GROUP	Lists the access groups to which you belong. (Internal)
LIST_LIMITS	Lists the various attributes that limit your command environment. (Internal)
LIST_MINI_COMMANDS	Lists the PRIMOS commands that you can use when you reach mini-command level. (Internal)
LIST_PRIORITY_ACCESS	Lists the priority ACL on your current disk. (Internal)

LIST_QUOTA	Gives information on records used and maximum records available in your current directory. (Internal)
LIST_REMOTE_ID	Lists your current set of remote IDs. (Internal)
LIST_SERVER_NAMES	Lists the active server names on the local system. (Internal)
LIST_SESSIONS	Lists the active ISC sessions on the local system. (Internal)
LIST_VAR	Lists the contents of an active global variable file. (Internal)
SIZE	Displays the size of files and the number of entries in directories, segment directories, and access categories. (External)
STATUS ME	Lists your user ID, user number, line number, and assigned devices. (Internal)

Information on Program and Command Execution

BATCH	Provides information on progress of user's batch jobs. (External)
DUMP_STACK	Produces a call/return trace of the user's stacks. (Internal)
LIST_EPF	Lists information on your EPFs. (Internal)
LIST_LIBRARY_ENTRIES	Lists the entrypoints in library EPFs. (Internal)
LIST_SEARCH_RULES	Lists the contents of all of your search lists. (Internal)
LIST_SEGMENT	Lists the segments you are using. (Internal)
PM	Lists the contents of the RVEC vector. (Internal)
PRERR	Gives locations and messages from PRIMOS' error vector, ERRVEC. (Internal)
TIME	Displays accounting information: time since login, CPU time used, and I/O time used. (Internal)

Sending Messages

MESSAGE	Sends a message from one terminal to another. (Internal)

Terminal I/O Handling

ASRCWD	Recovers a diverted terminal line from a serial printer, card reader, or terminal. (Internal)
RSTERM	Empties the terminal's input and/or output buffers. (Internal)
TYPE	Prints text at the terminal or into a command output file. (Internal)

I/O Device Handling

General Commands:

ASSIGN	Gives the user at the terminal control of a magnetic tape unit or other peripheral device. (Internal)
SET_ASYNC	Configures an asynchronous line connected to an AMLC, ICS, or LTS controller. (External)
UNASSIGN	Releases control of a previously assigned peripheral device. (Internal)

Magnetic Tapes:

ARCHIVE	Copies files and directories from disk to tape. (External)
ARCHIVE_RELEASE	Releases a tape created by ARCHIVE so that it can be reused. (External)
ARCHIVE_RESTORE	Restores files or directories from a tape created by ARCHIVE. (External)
GENERATE_CATALOG	Recreates a damaged catalog of a tape created by ARCHIVE or BACKUP or verifies the catalog's accuracy. (External)
LABEL	Creates an ANS COBOL level 1 volume label, or an IBM 9-track EBCDIC, or a 7-track BCD label on a magnetic tape. (External)
LIST_CATALOG	Lists the contents of a tape catalog created by ARCHIVE or BACKUP. (External)
LIST_TAPE	Lists information about file system objects recorded on a tape created by ARCHIVE, BACKUP, or TRANSPORT. (External)
MAGNET	Writes files or directories to tape or restores them from tape to disk. (External)
MAGRST	Restores files or directories from a tape created by MAGSAV. (External)
MAGSAV	Copies files or directories from disk to tape. (External)
TRANSPORT	Copies files or directories from disk to tape for transporting to another Prime site. (External)
TRANSPORT_RELEASE	Releases a tape created by TRANSPORT so that it can be reused. (External)
TRANSPORT_RESTORE	Restores files and directories from a tape created by TRANSPORT. (External)
UX_TAPE	Reads and writes tapes in a format suitable for communication with UNIX®. (External)

Punched Cards:

CPMPC	Punches the contents of a file onto a card deck. (External)
CRMPC	Writes the contents of punch cards into a file. (External)

Printers:

PRMPC	Prints a file on an MPC parallel interface printer. (External)
PRSER	Prints a file on a serial interface printer. (External)
PRVER	Prints a file on a configured printer/plotter. (External)

Communications

DPTXMTR	Provides information about the DPTX queues and events on the communication lines. (External)
DROPDTR	Drops the Data Terminal Ready (DTR) signal associated with an asynchronous line. (Internal)
FTR	Invokes the File Transfer Request utility to transfer files to or from a remote site. (External)
HDXSTAT	Displays information about the current status of all lines and sites of a half-duplex (HDX) network configuration. (External)
MONITOR_NET	Monitors the functioning of a network node. (External)
NETLINK	Connects the user to another system on the network. (Internal)
NTS_LINE	Places a terminal connected to a LAN Terminal Server (LTS) into Command mode. (Internal)
NTS_LIST_ASSOCIATE	Displays current associations (if any) of assignable NTS PRIMOS line numbers with their corresponding LTS name or line numbers. (Internal)
OWLDSC	Allows an OWL-1200 terminal to emulate an IBM 3277 Model 2 display station on systems where DPTX/DSC is running. (External)
PRTDSC	Invokes the printer emulation program on systems where DPTX/DSC is running. (External)
PT45DSC	Allows a PT45™ terminal to emulate an IBM 3277 Model 2 display station on systems where DPTX/DSC is running. (External)
PT46DSC	Allows a PT46™ terminal to emulate an IBM 3277 Model 2 display station on systems where DPTX/DSC is running. (External)

PTDSC	Allows a PST 100™ or a PT200™ terminal to emulate an IBM 3277 Model 2 display station on systems where DPTX/DSC is running. (External)
RJQ	Invokes the Remote Job Entry utility to transmit files to remote computer sites. (External)
SNADSC	Invokes the PRIME/SNA™ Interactive Terminal Emulation program that allows a supported Prime terminal to access applications on a remote IBM host. (External)
TCF	Invokes DPTX/TCF on a system where DPTX/TSF and DPTX/OSC are running, allowing you to access a remote IBM host computer. (External)
TRAMLC	Transmits or receives a file over an assigned asynchronous line between two Prime computer systems. (External)

Data Management

DBMS Subsystems:

CBLDML	Invokes the COBOL 74 data manipulation language (DML). (External)
CBLSUBS	Invokes the COBOL 74 DBMS subschema. (External)
CLUP	Invokes the DBMS Cleanup Processor. (External)
DBACP	Invokes the Database Administrator command process. (External)
DBUTL	Invokes a database dump utility that allows you to monitor the contents of a database schema and shared user table. (External)
DISCOVER	Invokes the DBMS and PRISAM™ query language and report writer. (External)
F77DML	Invokes the FORTRAN 77 DML preprocessor. (External)
FDML	Invokes the FORTRAN DML preprocessor. (External)
F77SUBS	Invokes the FORTRAN 77 DBMS subschema. (External)
FSUBS	Invokes the FORTRAN DBMS subschema. (External)
SCHDEC	Invokes the DBMS Schema Decompiler. (External)
SCHED	Invokes the Schema Editor (SCHED). (External)
SCHEMA	Invokes the DBMS schema Data Description Language compiler. (External)

FORMS:	
FAP	Invokes the FORMS Administrative Processor. (External)
FDL	Invokes the FORMS Definition Language (FDL) compiler. (External)
FED	Invokes the FORMS Editor. (External)
Prime INFORMATION:	
INFO	Invokes the Prime INFORMATION™ system. (External)
MIDASPLUS:	
CREATK	Invokes a program to build a template for a MIDASPLUS™ file. (External)
KBUILD	Invokes a program to build a keyed-index or direct-access MIDASPLUS file. (External)
KIDDEL	Invokes a program to delete all or part of the records in a MIDASPLUS file. (External)
MDUMP	Invokes a utility that dumps a MIDASPLUS file into a sequential disk file. (External)
MPACK	Invokes a utility that packs and restructures MIDASPLUS files. (External)
MPLUSCLUP	Invokes the MIDASPLUS cleanup utility. (External)
SPY	Invokes a utility that displays the information that MIDASPLUS uses. (External)
PRISAM:	
DIAG	Invokes a facility that verifies the structural validity of both relative and indexed PRISAM files. (External)
DISCOVER	Invokes the DBMS and PRISAM query language and report writer. (External)
FAU	Invokes a utility that creates, deletes, and manipulates PRISAM files. (External)

OAS (Office Automation System)

ATM	Logs you in to the OAS Advanced Text Management Option Selection Menu. (External)
OAS	Starts up the Prime Office Automation System. (External)
PTELE	Invokes the OAS Telephone Inquiry function. (External)
WORD	Invokes the PRIMEWORD™ Word Processing System. (Internal)

CAD/CAM

MEDUSA	Brings up a PRIME MEDUSA™ workstation. (External)
MEDCONFIG	Configures a PRIME MEDUSA installation. (External)
MEDUTIL	Runs PRIME MEDUSA utility programs. (External)
PLOT	Plots a metafile or device-specific plot file. (External)
THEMIS	Invokes a logic simulation program that allows you to interactively examine digital circuit models. (External)

System Settings

SVCSW	Controls handling of SVC instructions. (Internal)
VRTSSW	Sets the virtual sense switches. (Internal)

Summary of Operator Commands

The list below contains the PRIMOS commands used primarily by the System Administrator or operator. Many of these commands can be issued only from the supervisor terminal. These commands are explained briefly in Chapter 2. References to more detailed information are given at the entries for the commands.

ADDISK
ADMIN_LOG
AMLC
ATM_ADMIN
BACKUP
BACKUP_RELEASE
BACKUP_RESTORE
BATCH
BATGEN
BOOT_CREATE
CAB
CHAP
COMM_CONTROLLER
CONFIG
CONFIG_DSM
CONFIG_NET
CONFIG_NTS
CONFIG_UM
CONVERT_ENV
CONVERT_TO_ACLS
COPY_DISK
CRASH_AUDIT
DEVICE_ACLS
DISKS
DISPLAY_LOG
DISTRIBUTE_DSM
DPTCFG
DPTX
DUMP_SEGMENT
DUMP_USER
ELIGTS
FIND_RING_BREAK
FIX_DISK
FTGEN
FTOP
IROAM
LAB
LIST_ASSIGNED_DEVICES
LIST_ASYNC
LIST_COMM_CONTROLLERS
LIST_CONFIG
LIST_DISKS
LIST_DUMP
LIST_LAN_NODES
LIST_LHC_STATUS
LIST_LTS_STATUS
LIST_MEMORY
LIST_PRIMENET_LINKS
LIST_PRIMENET_NODES
LIST_PRIMENET_PORTS
LIST_PROCESS
LIST_SEMAPHORES
LIST_SERVER_NAMES
LIST_SESSIONS
LIST_SYNC
LIST_UNITS
LIST_VCS
LOGPRT
LOOK
LOOPBACK
MAKE
MAXSCH
MAXUSR
MIRROR_OFF
MIRROR_ON
NET
NTS_ASSOCIATE
NTS_UNASSOCIATE
OA_ADMIN
OPRPRI
PASSWORD_DIRS
PHYRST
PHYSAV
PLOTOP
PRIMOS
PRINT_SECURITY_LOG
PROP
REMOVE_PRIORITY_ACCESS
REPLY
RESET_DUMP
RESUS
RJOP
ROSAU
SECURITY_MONITOR
SECURITY_STATUS
SET_PRIORITY_ACCESS
SET_TIME_INFO
SETIME
SETMOD
SHARE
SHUTDN
SNA_3270
SNA_3270_CONFIG
SNA_PRINT
SNA_SERVER
SNA_SERVER_CONFIG
START_DSM
STATUS
STATUS_DSM
START_ISCNSR
START_NET
START_NTS
STARTUP
STOP_DSM
STOP_NET
STOP_NTS
TRANSFER_LOG
USRASR
WP_ADMIN

Summary of Command Functions

In the following summary, the PRIMOS command functions are ordered according to their type. The functions are described in greater detail in Chapter 3. Command functions and their arguments must be enclosed in square brackets when specified.

Arithmetic Functions

CALC	Evaluates arithmetic or logical expressions.
HEX	Converts a hexadecimal number to its decimal equivalent.
MOD	Divides one number by another and returns the remainder.
OCTAL	Converts an octal number to its decimal equivalent.
TO_HEX	Converts a decimal number to its hexadecimal equivalent.
TO_OCTAL	Converts a decimal number to its octal equivalent.

File System Functions

ATTRIB	Returns information (type, length, or date last modified) about a specified file or directory.
DIR	Returns the directory portion of a pathname.
ENTRYNAME	Returns the entryname (final component) portion of a pathname.
EXISTS	Determines whether a file system object exists and whether its file type matches the type specified.
GVPATH	Returns the pathname of an active global variable file.
OPEN_FILE	Opens a file for reading or writing.
PATHNAME	Returns the full pathname of a file system object.
READ_FILE	Reads a line from an ASCII file.
WILD	Produces a list of all names within a directory that match one or more wildcard names.
WRITE_FILE	Writes a line of text into an ASCII file.

String-handling Functions

AFTER	Returns the portion of a string that appears after some specified character(s).
BEFORE	Returns the part of a string that precedes some specified character(s).
INDEX	Returns the starting position of a specified substring within a string.
LENGTH	Returns the number of characters in a given string.
NULL	Tests for null strings.
QUOTE	Places a pair of quotation marks around a string, and doubles any quotation marks appearing within the string.

SEARCH	Compares two strings and returns the position of the first character in the first string that matches any character in the second string.
SUBST	Replaces text in one string with text from another.
SUBSTR	Returns a substring (specified by length and starting position) of a string.
TRANSLATE	Replaces character(s) in one string with character(s) from another.
TRIM	Removes characters from the left, right, or both sides of a specified string.
UNQUOTE	Removes outer quotation marks from around a specified text string and changes double quotation marks within the string to single quotation marks.
VERIFY	Compares two strings and returns the position of the first character in one that does not match any character in the other.

Miscellaneous Functions

ABBREV	Expands the value of an abbreviation or returns the pathname of the abbreviation file.
CND_INFO	Allows a CPL condition handler to examine the condition information of the most recent condition on the stack.
DATE	Displays the current date and/or time in a variety of formats.
GET_VAR	Returns the value of a named variable.
QUERY	Prints the contents of a specified text on the terminal screen and waits for a YES or NO reply.
RESCAN	Removes one level of quotation marks from a specified string and evaluates any function calls or variable references that no longer appear in quotation marks.
RESPONSE	Prints a specified prompt at the terminal and waits for any reply.

Summary of Changes in Commands

The following three sections summarize commands that are new, altered, or obsolete since the last edition of this guide at Rev. 21.0.

New Commands

The following commands have been added to this book at Rev. 22.0:

CAB
EDIT_CMD_LINE
IPSD
LAB
LEM
LIST_CONTIGUOUS_BLOCKS
LIST_SERVER_NAMES
LIST_SESSIONS
PRIMAN
PRIMON
START_ISCNSR
START_NSR
SYNCSORT
UX_TAPE

Altered Commands

The following commands have been altered at Rev. 22.0:

ADDISK
ARCHIVE
ASSIGN
BACKUP
BASICV
BIND
CBL
CHAP
CONFIG_NET
COPY_DISK
DBACP
DBUTL
EDIT_PROFILE
ELIGTS
FIX_DISK
INITIALIZE_COMMAND
_ENVIRONMENT
LABEL
LIST_ASYNC
LIST_ASSIGNED_DEVICES
LIST_DISKS
LIST_PRIMENET_NODES
LIST_PRIMENET_PORTS
LIST_PROCESS
LIST_SYNC
LIST_UNITS
LIST_VCS
MAGSAV
MAGRST
MAKE
MAXSCH
MIRROR_ON
PHYRST
PROP
SET_ASYNC
SET_RBF
SPOOL
SPY
START_DSM
START_NET
STATUS
STOP_NET

Removed Commands

The following commands have been removed at Rev. 22.0:

CDML
COBOL
CSUBS
EVENT_LOG
FIXRAT
NCOBOL
POWER
PRINT_NETLOG
PRINT_SYSLOG
RPG

2 Dictionary of PRIMOS Commands

▶ A

See ATTACH.

▶ ABBREV { pathname [options] / options }

ABBREV defines user abbreviations.

A user abbreviation is an abbreviation that you create for one or more PRIMOS commands and their arguments. You can then use these abbreviations during interactive sessions and in CPL programs that use the &EXPAND directive.

To use abbreviations, you must follow these procedures:

1. Create an abbreviation file.
2. Define abbreviations. The abbreviations are stored in the abbreviation file, which can hold approximately 200 abbreviations.
3. Activate the abbreviation file to use the abbreviations stored in it. You can have more than one abbreviation file, but only one can be active at any given time.
4. If you are in a multiuser environment, deactivate the abbreviation file at the end of your work session.

When an abbreviation file is active, the PRIMOS abbreviation preprocessor scans each command line you entered at your terminal. (You cannot use abbreviations in a command input file.) The abbreviation preprocessor checks each token (that is, each

word or number) against the abbreviation file. When the preprocessor finds an abbreviation, it expands the abbreviation to its full form and then passes it to the standard command preprocessor for execution.

Note

You can use abbreviations only if the System Administrator has enabled the abbreviation preprocessor.

Creating an Abbreviation File

To create an abbreviation file, use this command format:

ABBREV new-pathname –CREATE

new-pathname is the name of your abbreviation file. Use a filename if the file will be in your current directory. The command also activates the file. If you had an active abbreviation file when you issued the command, that file is deactivated and the new file is activated.

The example below shows the creation of an abbreviation file in the current ACL directory:

```
OK, ABBREV RBJ.ABBREV -CREATE
Creating new abbreviation file: <DPT2>RBJ>RBJ.ABBREV (ab_file_)
OK,
```

If the directory in which the file is created has a password, the password must be specified after the directory name and both must be enclosed in single quotation marks, as shown here:

```
OK, AB 'DM READY'>DM.ABBREV -CREATE
Creating new abbreviation file: <STD2>DM READY>DM.ABBREV (ab_file_)
OK,
```

In this example, READY is the password for the directory DM. The abbreviation file, DM.ABBREV, is placed in DM.

Defining Abbreviations

To define an abbreviation, use this command format:

ABBREV [pathname] { –ADD | –ADD_ARGUMENT | –ADD_COMMAND } name value

You do not have to specify **pathname** (the name of the abbreviation file) if the file is active. If you specify **pathname**, the command also activates the file.

name is the name of the abbreviation and **value** is what it stands for. Each abbreviation must have a name and a value. **name** and **value**, as well as the three ADD options, are defined below.

Abbreviation Names

Abbreviation names can be as many as eight ASCII characters in length, but cannot use spaces, single quotation marks ('), commas, right angle-brackets (>), and vertical bars (|). You should not begin names with hyphens because of possible confusion with PRIMOS command options. You can specify the name in lowercase or uppercase characters because the abbreviation preprocessor converts all lowercase characters to uppercase.

In the following example, WHO is the abbreviation name:

```
OK, ABBREV -ADD_COMMAND WHO STATUS USERS
OK,
```

This example creates an abbreviation named WHO whose value is STATUS USERS. Entering WHO on the command line has the same effect as entering STATUS USERS.

Abbreviation Values

The value of an abbreviation can be any ASCII character string, including spaces. Leading spaces and/or tabs are removed during definition. The allowance of spaces means that the value can consist of several tokens. (A **token** is any word, number, or variable.) In the previous example, the value of the abbreviation WHO consists of two tokens, STATUS and USERS.

You can define as many as nine variables in **value** by enclosing a number within percent signs (for example, %1%). Variables allow you to specify one or more words (or numbers) after the abbreviation and have those words inserted in place of the variables. For example, the following command defines an abbreviation named CAN which has three variables:

```
OK, ABBREV -ADD_COMMAND CAN SPOOL -CANCEL %1% %2% %3%
OK,
```

You can now use CAN on the command line to cancel a maximum of three spooled files. The first number you give after CAN is inserted in place of the variable %1%, the second number is inserted in place of %2%, and the third number is inserted in place of %3%. Thus, the command

```
OK, CAN 2 4 7
```

is expanded to

```
OK, SPOOL -CANCEL 2 4 7
```

If you supply fewer words or numbers than there are variables, the extra variables are ignored. For example, using the above abbreviation, the command

```
OK, CAN 4
```

is expanded to

```
OK, SPOOL -CANCEL 4
```

Variables %2% and %3% are ignored.

If you supply more words or numbers than there are variables, the extra words or numbers are added to the end of the command line. This may cause errors in certain commands.

The –ADD_COMMAND Option

The –ADD_COMMAND option creates a command abbreviation. Command abbreviations are expanded only when you use them in the command position of the command line (that is, immediately after the OK, prompt). Thus, you can use command abbreviations as if they were PRIMOS commands.

The two examples above (WHO and CAN) are command abbreviations. Here is another example of defining a command abbreviation:

```
OK, ABBREV -ADD_COMMAND CP PASCAL %1% -NO_MAP -LISTING
OK,
```

CP is the name of the abbreviation. Entering CP and a filename at the PRIMOS command line (for example, CP PROG.PASCAL) is equivalent to entering the PRIMOS command

```
OK, PASCAL PROG.PASCAL -NO_MAP -LIST
```

When you list the contents of an abbreviation file with the ABBREV –LIST command, command abbreviations are preceded by a (C) symbol, as in the next example:

```
OK, ABBREV -LIST
Abbreviation file: <DPT2>RBJ>RBJ.ABBREV
Abbreviations: 3

(C) CAN       SPOOL -CANCEL %1% %2% %3%
(C) CP        PASCAL %1% -NO_MAP -LISTING
(C) WHO       STATUS USERS

OK,
```

The –ADD_ARGUMENT Option

The –ADD_ARGUMENT option creates an argument abbreviation. Argument abbreviations are expanded only when you use them as arguments to a PRIMOS command or to a command abbreviation.

One example of an argument abbreviation is to define an abbreviation for a directory that you often use, as follows:

```
OK, abbrev -add_argument proj <staff>tom>project
OK,
```

You could then use this abbreviation to the ATTACH command

```
OK, ATTACH PROJ
```

This command line would be expanded to

```
OK, ATTACH <STAFF>TOM>PROJECT
```

When you list the contents of an abbreviation file with the ABBREV –LIST command, argument abbreviations are preceded by an (A) symbol, as in the next example:

```
OK, ABBREV -LIST
Abbreviation file: <ADMIN>TRB>TRB.ABBREV
Abbreviations: 1

(A) PROJ      <staff>tom>project

OK,
```

The –ADD Option

The –ADD option creates an abbreviation that is expanded regardless of where you use it on the command line. Therefore, it is possible to define an abbreviation that you could use either as a command or as an argument to a command.

The use of the –ADD option, however, is discouraged because of the possible confusion of the abbreviation, when expanded, with PRIMOS commands or with the names of file system objects. For more information on the differences between –ADD and either –ADD_ARGUMENT or –ADD_COMMAND, see the *PRIMOS User's Guide*.

Activating an Abbreviation File

At login, your abbreviation file is not automatically activated. You must activate the file yourself by issuing the appropriate ABBREV command interactively or in a login CPL program or command input file.

Use the command format below the first time you activate the file during a work session:

ABBREV pathname

pathname is the name of the abbreviation file. Use a filename if the file is in your current directory.

If later during the session, the file is deactivated by you or by a system error, reactivate the file with the above format or with the command

ABBREV [pathname] –ON

If you do not specify **pathname**, the command activates the file you were previously using.

Deactivating an Abbreviation File

Your abbreviation file is automatically deactivated when you log out. To deactivate the file during a work session, issue the command

ABBREV –OFF

If you are in a multiuser environment, you should deactivate your abbreviation file at the end of your work session by issuing the ABBREV –OFF command or by logging out.

ABBREV Options

All of the ABBREV options are listed below. In each case, supplying the name of the abbreviation file within the command activates the file as well as performing the option-related task. If the file is already active, the filename is not needed.

Option	*Meaning*
–ADD name value	Adds the abbreviation **name** to the current file with the meaning **value**. If **name** already exists, you are asked if it should be replaced. The abbreviation is expanded when used in either the command position or argument position of a command line.

–ADD_ARGUMENT name value	Adds the abbreviation **name** to the current file with the meaning **value**. If **name** already exists, you are asked if it should be replaced. The abbreviation is expanded only when used in the argument position of a command line.
–ADD_COMMAND name value	Adds the abbreviation **name** to the current file with the meaning **value**. If **name** already exists, you are asked if it should be replaced. The abbreviation is expanded only when used in the command position of a command line.
–CHANGE name-1 [...name-n]	Changes the specified abbreviations so that they are expanded anywhere on the command line.
–CHANGE_ARGUMENT name-1 [...name-n]	Changes the specified abbreviations to argument abbreviations, so that they are expanded only in the argument position on the command line.
–CHANGE_COMMAND name-1 [...name-n]	Changes the specified abbreviations to command abbreviations, so that they are expanded only in the command position of the command line.
–CHANGE_NAME old-name new-name	Changes the name of the abbreviation **old-name** to **new-name**.
–CREATE	Creates and activates an empty abbreviation file. You must supply a pathname or filename. If the file already exists, the command activates that file.
–DELETE name-1 [...name-n]	Deletes the specified abbreviations from the abbreviation file.
–EXECUTE rest-of-line	Passes the rest of the command line to the command processor for execution without first expanding it. Must be the last option on the command line because all text that follows (except comments) is assumed to be the argument.
–EXPAND rest-of-line	Expands the rest of the command line and displays it at the terminal, but does not execute it. Must be the last option on the command line because all text that follows (except comments) is assumed to be the argument.
–EXPAND_EXECUTE rest-of-line	Expands the rest of the command line and then passes it to the command processor for execution. Must be the last option on the command line because all text that follows (except comments) is assumed to be the argument.
–HELP	Displays a summary of the ABBREV options.
–LIST [name-1 [...name-n]]	Lists the specified abbreviations or, if no names are specified, all abbreviations from the current abbreviation file. The names can be wildcard names. When listed, abbreviations created by –ADD_COMMAND are preceded by (C), and those created by –ADD_ARGUMENT are preceded by (A).
–NO_QUERY	Replaces the old abbreviation without asking you for confirmation. (Useful only if followed by one of the ADD options.)

–NO_VERIFY	Disables Verify mode so that an expanded command line is not displayed before execution. (Default mode.)
–OFF	Deactivates the abbreviation file and turns off abbreviation expansion.
–ON	Activates the abbreviation file and turns on abbreviation expansion. If a pathname is given, activates that file. If a pathname is not given, reactivates the file previously in use.
–STATUS	Displays the pathname of the current abbreviation file and the number of abbreviations it contains.
–VERIFY	Enables Verify mode, which expands any abbreviations on the command line and then displays the entire command line before executing it. If the command line contains no abbreviations, the line is still displayed before execution.

Further Facts About Abbreviations

The following is additional information about the creation and use of abbreviations:

- If a user-defined abbreviation is identical to a PRIMOS-defined abbreviation, the user's abbreviation takes precedence when the user's abbreviation file is activated. The PRIMOS abbreviation is therefore unavailable to the user during that time, unless it is enclosed in quotation marks and typed in uppercase.
- Abbreviations cannot be defined for subcommands.
- Abbreviations may be used as arguments to other abbreviations. When this happens, the abbreviations are treated as though they were nested. The innermost abbreviation is expanded first, and any variables needed by it are used. The expanded expression is then available to the next abbreviation, together with any other (unused) tokens in the command line, and so on.
- Abbreviations may be defined for ABBREV commands. The abbreviations preprocessor checks all command lines after expanding the first token. If the first token expands to ABBREV, the rest of the line is read without expansion. Examples of these handy abbreviations are .A for ABBREV –ADD_COMMAND and .L for ABBREV –LIST.
- To include a command function in an abbreviation, use the tilde (~) syntax-suppression character before the ABBREV command. For example, the following command creates an abbreviation (W) that uses the TYPE command and the DIR and PATHNAME command functions to display the pathname of your current directory:

```
OK, ~AB -AC W TYPE You are at [DIR [PATHNAME*]]
```

For further information on the use of the tilde as a syntax-supression character, see the section titled Syntax Suppression in Chapter 4.

- A wildcard may be used with the ABBREV command. When you use a wildcard in place of a name, ABBREV selects all abbreviation names that match the wildcard. For example, you can list all abbreviation names beginning with a period (.) with the command

```
ABBREV -LIST .@
```

For further information on wildcards, see Chapter 4 of this guide.

- Tokens on a command line (that is, abbreviations, commands, or arguments) may be separated by spaces, tabs, commas, or right angle-brackets (>).
- Characters placed in quotes within a command line are not expanded by the abbreviation preprocessor, but are handed on literally — quotes and all — to the command processor.

Example of Creating Abbreviations

An example of creating and using abbreviations is shown below. For the sake of clarity, the example uses Verify mode (which is enabled by the ABBREV –VERIFY command) to make PRIMOS echo the expanded abbreviations before using them.

```
OK, ABBREV LOGIN.ABBREV -CREATE
Creating new abbreviation file: <FOREST>BEECH>LOGIN.ABBREV (ab_file_)
OK, ABBREV -ADD U <FOREST>BEECH>%1%
OK, ABBREV -ADD SP SPOOL %1% -ATT PRINTA %2% %3% -LIST
OK, ABBREV -LIST
Abbreviation file: <FOREST>BEECH>LOGIN.ABBREV
Abbreviations: 2

    SP        SPOOL %1% -ATT PRINTA %2% %3% -LIST
    U         <FOREST>BEECH>%1%

OK, ABBREV -VERIFY
OK, SP U UPDATE.MEMO -DEFER 2200
(listen_) "SPOOL <FOREST>BEECH>UPDATE.MEMO -ATT PRINTA -DEFER 2200 -LIST"
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Request 21 added to queue, 1 records : <FOREST>BEECH>UPDATE.MEMO

System
Request  Time  User            File                              No Size  State
-------- ----- --------------- --------------------------------- -- ----  -----
21       1450  BEECH           UPDATE.MEMO                       1  1     Defer
OK,
```

U is expanded first because it is the inner abbreviation. It wants one variable (%1%) so it takes UPDATE.MEMO as that variable. When SP is expanded, it takes the expanded value of U (which is <FOREST>BEECH>UPDATE.MEMO) as its variable %1% and then takes –DEFER and 2200 as its variables %2% and %3%.

The next part of the example adds two abbreviations, .A and .L, for the commands ABBREV –ADD_COMMAND and ABBREV –LIST:

```
OK, AB -VERIFY
(listen_) "AB -VERIFY"
OK, AB -NO_VERIFY
(listen_) "AB -NO_VERIFY"
OK, AB -VERIFY
OK, AB -AC .A ABBREV -AC
(listen_) "AB -AC .A ABBREV -AC"
OK, .A .L ABBREV -LIST
(listen_) "ABBREV -AC .L ABBREV -LIST"
OK, .L
(listen_) "ABBREV -LIST"
Abbreviation file: <FOREST>BEECH>LOGIN.ABBREV
Abbreviations: 4

(C) .A        ABBREV -AC
(C) .L        ABBREV -LIST
    SP        SPOOL %1% -ATT PRINTA %2% %3% -LIST
    U         <FOREST>BEECH>%1%

OK,
```

The (C) symbol in the abbreviation list indicates that the abbreviation defines a command.

For more information on the ABBREV command and on user abbreviations, see the *PRIMOS User's Guide*.

▶ ADDISK { [PROTECT] pdev-1 [...pdev-n] | pdev –RENAME diskname | diskname-1 [...diskname-9] –ON nodename } *Operator Command*

ADDISK starts up a local or remote disk on the system. For details, see the *Operator's Guide to System Commands*.

▶ ADD_REMOTE_ID remote-id [password] –ON nodename [–PROJECT project-id] [–PROMPT]

ADD_REMOTE_ID specifies the remote user ID to be used on a remote system.

When you work on a remote system, a type of phantom called a slave does your work using your local user ID. However, before you can access a remote system configured to force user validation, you must specify a valid remote ID from your system with ADD_REMOTE_ID.

If your local user ID is valid on a remote system but you do not want the slave to use it, you can also have a remote user ID. A remote ID may allow you certain ACL rights that your local user ID does not.

In both of the above cases, the remote ID must have been created for you by the System Administrator of the remote system. When you issue the ADD_REMOTE_ID command, the remote ID is added to a list that can hold a maximum of 16 remote IDs. Each remote ID is valid on a separate remote system and you cannot specify more than one remote ID for any one remote system. To list your remote IDs, issue the LIST_REMOTE_ID command.

If you add a remote ID to a system that already has one, the new remote ID replaces the old ID. The remote IDs exist until you log out or until you remove them from the list of remote IDs with the REMOVE_REMOTE_ID command.

Arguments and Options

The –PROJECT option specifies a project ID, which is required on some systems. The –PROMPT option sets the terminal to half duplex and prompts you for your password.

The arguments for ADD_REMOTE_ID are the same as those for the LOGIN command. For an explanation of naming rules for the user ID, password, node name, and project ID, see the LOGIN command.

Example of Using ADD_REMOTE_ID

In the following example, a person with a user ID of LAURA on SYS1, her local system, has a valid remote ID of LAURA2 on SYS2. The remote ID has a password of BLUEJAY and is affiliated with the project GRAND. To activate the remote ID LAURA2, LAURA would type the following from her terminal on SYS1:

```
ADD_REMOTE_ID LAURA2 BLUEJAY -ON SYS2 -PROJECT GRAND
```

For further information on remote IDs, see the *User's Guide to Prime Network Services*. *See also* LIST_REMOTE_ID; LOGIN; REMOVE_REMOTE_ID.

▶ ADMIN_LOG logname [log_type] [options]

Operator Command

ADMIN_LOG creates and administers Distributed System Management (DSM) logs. User access to ADMIN_LOG is controlled by the System Administrator.

DSM provides networked systems management from any convenient point on the network. For further information, see the *DSM User's Guide*.

▶ ARCHIVE pathname –MT n –VOLID tapename [options]

ARCHIVE copies file system objects from disk to tape. File system objects include files, directories, access categories, and Recovery Based Files.

pathname is the file or directory to be copied to tape. **–MT n** specifies the unit number of the tape drive on which the ARCHIVE tape is mounted. **tapename** is the volume name of the ARCHIVE tape on which the objects are to be stored. The ARCHIVE tape can be a current tape or a new tape.

Before you can use ARCHIVE, you must assign the tape drive unit with the ASSIGN command and mount the correct reel on the drive.

To display online information on the syntax and options of the ARCHIVE command, use the command format

ARCHIVE –HELP [option]

For detailed information on the options and use of ARCHIVE, see the *Data Backup and Recovery Guide.*

See also ARCHIVE_RESTORE; ARCHIVE_RELEASE; GENERATE_CATALOG; LIST_CATALOG; LIST_TAPE.

▶ ARCHIVE_RELEASE –VOLID tapename [options]

ARCHIVE_RELEASE releases an ARCHIVE tape reel so that its data can be overwritten.

tapename identifies the reel to be released. The tape must have been created with the ARCHIVE command. Only the tape owner or a user with access rights to the tape catalog can release the tape. If the tape catalog does not exist, only the tape owner or the System Administrator can release the tape.

Before using ARCHIVE_RELEASE, you must assign the tape drive with the ASSIGN command and mount the correct tape on the drive.

To display online information on the syntax and options of the ARCHIVE_RELEASE command, use this command format:

ARCHIVE_RELEASE –HELP [option]

For detailed information on the options and use of ARCHIVE_RELEASE, see the *Data Backup and Recovery Guide.*

See also ARCHIVE; ARCHIVE_RESTORE; GENERATE_CATALOG; LIST_CATALOG; LIST_TAPE.

▶ ARCHIVE_RESTORE pathname [new-pathname] –MT n [options]

ARCHIVE_RESTORE copies file system objects from an ARCHIVE tape to disk.

After you save file system objects on tape with the ARCHIVE command, you can restore the data back to disk with ARCHIVE_RESTORE. File system objects include files, directories, access categories, and Recovery Based Files. Before using ARCHIVE_RESTORE, you must assign the tape drive with the ASSIGN command and mount the correct reel.

To display online information on the syntax and options of the ARCHIVE_RESTORE command, use the command format

```
ARCHIVE_RESTORE –HELP [option]
```

For detailed information on the options and use of ARCHIVE_RESTORE, see the *Data Backup and Recovery Guide*.

See also ARCHIVE; ARCHIVE_RELEASE; GENERATE_CATALOG; LIST_CATALOG; LIST_TAPE.

▶ ARID

See ADD_REMOTE_ID.

▶ ASRCWD [number]

ASRCWD sets a virtual control word that selects the input or output device for diverted terminal I/O.

ASRCWD works only with serial I/O. The chief use of ASRCWD is to recover from the following situation: Output is being sent to a serial line printer when a program aborts or CONTROL-P is pressed. At this point, you cannot get output (from the Editor, for example) displayed at your terminal. To recover and resume output to the terminal, issue the following command:

```
ASRCWD 0
```

For further information on ASRCWD, see Appendix C of this guide.

▶ ASSIGN { device [–WAIT] | DISK pdev [–WAIT] | ASYNC –LINE n [–TO m] }

ASSIGN obtains complete control of a disk, line, or peripheral device from the user terminal.

device is an available device of the type listed below. The ASYNC, DISK, MT/MTX, and SMLC arguments are described in greater detail in following sections.

Device	*Meaning*
ASYNC –LINE n [–TO m]	Asynchronous communications line. See the section Assigning Asynchronous Lines below for explanation of options.
CARDR	Serial card reader.
CENPR	Serial printer.
CE2PR	Serial printer.
CRn	MPC parallel card reader or reader/punch. **n** is 0 or 1.
DISK pdev	Physical disk partition. **pdev** is the physical disk number, in octal.
GSn	Vector General graphics display terminal. **n** is from 0 to 3, inclusive.
MGn	Megatek graphics display terminal. **n** is from 0 to 3, inclusive.
MTpdn [options]	Magnetic tape unit. **pdn** is the physical device number, which ranges from 0 to 7, inclusive. **options** are described below.
MTX –ALIAS MTldn [options]	Magnetic tape unit. **ldn** is the logical device number, which ranges from 0 to 7, inclusive. **options** are described below.
PLOT	Printer/plotter.
PRn	Line printer. **n** is from 0 to 3, inclusive.
PTR	Paper tape reader.
PUNCH	Paper tape punch.
SMLCnn	Synchronous communications line. **nn** is from 00 to 07, inclusive.

The –WAIT option indicates that, if the specified device is not available, you can wait until it is free. If the device is currently assigned to another user and you did not specify –WAIT, the following message is displayed:

```
The device is in use.   (ASSIGN)
ER!
```

If you did specify –WAIT, your request is queued until the device is unassigned by another user or until the other user presses the CONTROL-P or BREAK keys or logs out. The terminal from which you invoked the ASSIGN command cannot be used until you press the CONTROL-P or BREAK keys or until the assigned device is available again for assignment.

The two examples below illustrate the use of the ASSIGN command with peripheral devices:

Example 1: To assign the serial line printer and queue the assignment if the printer is already assigned, enter the command

```
OK, ASSIGN CENPR -WAIT
```

Example 2: To assign the paper tape reader, enter the command

```
OK, ASSIGN PTR
```

If you do not assign a device and attempt to perform I/O to or from the device, the following error message is displayed:

```
Device not assigned.
ER!
```

Assigning Magnetic Tapes

For magnetic tape, the ASSIGN command either indicates by number which physical tape drive you want, or provides a description of a tape drive that meets your requirements. In addition, you can make special requests to the system operator with an ASSIGN option (for example, removing the write ring or mounting a particular tape). These requests are useful primarily for batch jobs.

The command format for assigning magnetic tapes is

```
ASSIGN { MTpdn [options]      }
       { MTX –ALIAS MTldn     }
```

The meaning of the two arguments are

Argument	*Meaning*
MTpdn	Assigns the magnetic tape drive specified by **pdn**. **pdn** is the physical device number assigned to each drive at system startup. **pdn** is from 0 to 7, inclusive.
MTX	Assigns any available drive. Must be accompanied by the –ALIAS MT**ldn** option. The drive assigned depends on the other options specified on the command line.

The options for either argument are

Option	*Meaning*
–ALIAS MTldn	Specifies a tape drive with a logical device number. **ldn** is from 0 to 7, inclusive. A logical device number is a user-specified number assigned to a particular physical drive unit. When used with the MT**pdn** argument, **ldn** is mapped into **pdn** in subsequent magnetic tape operations.
–DENSITY bpi	Specifies the tape density in bytes per inch. **bpi** must be one of the following: 800, 1600, 3200, or 6250. This option sets tape density automatically on Version 3 tape drives (drives that can handle 6250 bpi), but requires operator intervention on Version 0, 1, and 2 drives so that density can be set manually.
–MOUNT	Instructs the operator to mount a new tape reel. This option is used in the middle of a mag tape procedure (such as MAGSAV or MAGRST) when one reel of tape is exhausted and another is needed. It is usually accompanied by the –TPID option. Unlike other uses of the ASSIGN command, use of the –MOUNT option requires that the drive has been assigned already by a previous ASSIGN command.
–RETENSION	Causes the tape to be fast-forwarded to the end of the tape and then rewound to the beginning of the tape. This procedure stabilizes the tape-to-head pressure and stacks the tape evenly on the reel. Used only with cartridge tape drives (Version 5).
–RINGOFF	Instructs the operator to remove the write ring from the tape so that it may be read but not written.
–RINGON	Instructs the operator to place the write ring on the tape so that it may be both read and written.
–SPEED { **25** / **50** / **100** }	Sets the tape speed at 25, 50, or 100 inches per second.

–TPID id	Instructs the operator to mount a particular reel of tape described by **id**. **id** is a list of tape identifiers (arguments) that refers to a specific reel of tape and/or type of tape drive (name, number, and so on). Identifiers cannot begin with a hyphen (-) and cannot contain the following delimiters: commas, spaces, .NL., and /*. Requires operator intervention.
–WAIT	Indicates the user will wait until the requested drive is available.
–7TRK	Specifies a 7-track tape drive. (Default is 9-track.) Like the –9TRK option, this option is usually used with the MTX argument. Requires operator intervention.
–9TRK	Specifies a 9-track tape drive. (Default) Like the –7TRK option, this option is usually used with the MTX argument. Requires operator intervention.

Examples of Assigning Magnetic Tapes

The following four examples illustrate the use of the ASSIGN command to assign tape drives.

Example 1:

```
OK, AS MT3
Device MT3 assigned.
OK,
```

Magnetic tape drive MT3 is assigned. (3 is the physical device number.)

Example 2:

```
OK, ASSIGN MT4 -ALIAS MT0
Device MT4 assigned.
OK,
```

Magnetic tape drive MT4 is assigned by means of a logical device number. Henceforth, MT4 is referred to as logical MT0. The physical-to-logical number correspondence can be listed with the STAT DEV command:

```
OK, AS MT4 -ALIAS MT0
Device MT4 assigned.
OK, STAT DEV

Device User name                Usrnum Ldevice
MT4    HENRY                      67    MT0
```

Example 3:

```
OK, assign mtx -alias mt4
Device MT2 Assigned.
OK,
```

The operator is requested to assign any available tape drive as logical device 4. In response to the command, the operator sends a message to your terminal that indicates which physical drive has been assigned. In this example, the operator assigned magnetic tape drive MT2 in response to the request.

Example 4:

```
OK, as mtx -alias mt3 -tpid power -9trk -ringoff -density 6250
Device MT0 Assigned.
OK,
```

The operator is requested to mount a 9-track tape named POWER on any drive that can handle 6250 bpi. In addition, the user wants write protection and an alias of MT3 to whatever device the operator chooses. The operator assigned tape drive MT0 in response to the command. If the request could not be handled by the operator, the following message would be displayed instead:

```
MagTape Assignment Request Aborted (ASSIGN)
ER!
```

Operator Intervention

The System Administrator uses the SETMOD command to determine the operator's role in tape assignments. Three modes are possible:

- Each user can assign a tape drive from any terminal, and operator intervention is necessary only for processing special requests. This is the default mode.
- Each user must send all assignment requests through the operator, who controls all access to tape drives. The operator then sends messages to the user terminal indicating the status of the assignment request.
- Tape-drive assignment from any user terminal is strictly forbidden. This feature is used to restrict access to tape drives in security-conscious environments, or to warn users when the operator is not available to process requests. In this mode, any attempt by a user to assign a magnetic tape drive results in the following message:

```
No MagTape Assignment Permitted.  (AS)
ER!
```

Assigning Disks

To assign a disk, use the command format

ASSIGN DISK pdev

pdev is the physical disk number. (Use the STATUS DISKS command to list the physical disk numbers of your system's disks.) The disk cannot be any of the following: the paging disk, a disk already assigned to another user, or a disk not in the Assignable Disks Table.

Before you can assign a disk, you must ask the operator to add the disk to the Assignable Disks Table. The operator prepares this table by using the DISKS command from the supervisor terminal. This restriction provides a degree of system integrity because it prevents users from assigning a disk without the system operator's knowledge, or from assigning disks or partitions the operator wishes to reserve for special use.

To assign a disk that has been started by ADDISK, the disk must first be shut down at the supervisor terminal by the SHUTDN command.

The maximum number of disk partitions that may be assigned to all users at any one time is 10. If an attempt is made to assign too many disks, the following message is displayed:

```
ASSIGN TABLE FULL
```

The following example assigns to the user the disk with the physical disk number 2260:

```
OK, ASSIGN DISK 2260
```

For further information on assigning disks, see the *Operator's Guide to System Commands* and the *Operator's Guide to File System Maintenance*.

Assigning Asynchronous Lines

You can assign an asynchronous line only if it has been configured to be assigned and if it is not assigned to another user.

At Rev. 21.0, the ASSIGN ASYNC command (decimal notation) replaces the ASSIGN AMLC command (octal notation). ASSIGN ASYNC allows you to assign either a line or a range of consecutive lines on one command line. To assign an asynchronous line, use the following command format:

ASSIGN ASYNC –LINE n [–TO m]

n is the decimal line number of the line to be assigned. If you are assigning more than one line, **n** is the first value in a series of line numbers. **–TO** assigns a

consecutive number of lines: **n** is the first number and **m** is the last number in the series of line numbers that you are assigning. The value of **m** must be greater than **n**.

To set assignable line characteristics, including terminal line characteristics and Network Terminal Service (NTS) lines, use the SET_ASYNC command, described later in this chapter. For further information on asynchronous lines, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

Assigning Synchronous Lines

To assign an SMLC line, use the following command format:

AS SIGN SMLC nn [-WAIT]

nn is an synchronous line number ranging from 00 through 07, inclusive. For details on configuring synchronous lines and maintaining synchronous controllers, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

Assigning NTS Lines

You can assign Network Terminal Service (NTS) lines provided that they have been associated by the system operator or the Network Administrator. The PRIMOS line number must be specified in decimal notation. NTS lines assigned with the ASSIGN command can be unassigned only with the UNASSIGN command. For further information, see the *NTS User's Guide.* For more information on NTS lines, see the *NTS Planning and Configuration Guide.*

▶ ATM

ATM displays the Advanced Text Management Option Selection Menu of the Prime Office Automation System (OAS). For detailed information, see the *OAS Advanced Text Management Guide.*

▶ ATM_ADMIN

Operator Command

ATM_ADMIN allows the Office Automation System (OAS) Administrator to create and maintain the document database and files relating to the Document Database Indexing function. For detailed information, see the *OAS System Administrator's Guide.*

ATTACH pathname

ATTACH moves the user's attach point from the current directory to another directory. Your **current directory** is the directory to which you are attached at any given moment. Use ATTACH to attach to another directory specified by **pathname**. Pathnames are specified as follows:

- To attach to a top-level directory, use only the directory name. For example:

```
OK, ATTACH BEECH
```

  If, however, two or more local disks have top-level directories named BEECH, you are attached to the BEECH directory on the disk that has the lowest logical device number. See the Search Order section below for details.

- To attach to a directory subordinate to your current directory, begin the pathname with an asterisk (*). For example:

```
OK, ATTACH *>BRANCH5
```

- To attach to a directory on a particular disk or to attach to the MFD of a disk, begin the pathname with the diskname. Here are two examples:

```
OK, ATTACH <FOREST>OAK>ACORN
OK, ATTACH <FOREST>MFD
```

  You can substitute the logical device number of the disk for the diskname. For example, if the logical device number of <FOREST> is 3, then the previous two ATTACH examples can be given as follows:

```
OK, ATTACH <3>OAK>ACORN
OK, ATTACH <3>MFD
```

  Use the STATUS DISKS command to obtain the logical device numbers of the disks connected to your system.

Note

An obsolete form of the ATTACH command also uses logical disk (ldisk) and key parameters as arguments. This version of ATTACH is documented in Appendix C, Old Commands.

Protection

To attach to an ACL directory, you must have Use (U) access to all directories in the pathname.

To attach to a password directory, you must supply the appropriate password following the directory name. For example, the following command attaches you to directory BEECH that has SECRET as its password:

```
OK, ATTACH BEECH SECRET
```

Search Order

If you do not specify a diskname as part of the pathname, all the disks to which you have access are searched. The disks are searched in the following order:

1. All local disks are searched in order of their logical device (ldev) numbers.
2. If your local system is on a network, remote disks are then searched in order of their logical device numbers.

If an error message is returned following an ATTACH command (for example, if an MFD is not found), you remain attached to the directory in which you were working when you issued the command.

The search rules may cause unexpected attachments if there are duplicate top-level directory names or duplicate MFD names. The following two examples illustrate this.

Example 1: A system has four local disks. The logical device number of disk <GRP1> is 1 and that of disk <GRP3> is 3. Both disks have top-level directories named STAFF. If you are on <GRP3> and want to attach to <GRP3>STAFF, the command:

```
OK, ATTACH STAFF
```

attaches you to <GRP1>STAFF instead of <GRP3>STAFF because PRIMOS first searches the disk with the lowest logical device number. In such cases, you must specify the diskname in addition to the directory name.

Example 2: Suppose two users, Robert and Mary, are on the same system. Robert has rights to the MFD <LOCAL> on that system while Mary has none. When Robert issues the command:

```
OK, ATTACH HOUSE
```

he is attached to <LOCAL>HOUSE on his local system. If Mary issues the same command, <LOCAL> is not searched because she has no rights to it. If the system is on a network, Mary might be attached to a directory called <FAROUT>HOUSE on a remote system because after searching all local disks, PRIMOS then searches remote disks.

Attach Examples

The examples below demonstrate typical attachments.

```
OK, attach <writer>shakespeare
OK, create plays
OK, a shakespeare>plays
OK, create (comedies tragedies)
OK, ld

<WRITER>SHAKESPEARE>PLAYS (ALL access)
1 record in this directory, 3 total records out of quota of 0.

2 Directories.

COMEDIES          TRAGEDIES

OK, a *>histories
Not found. HISTORIES (ATTACH)
ER! a *>6ragedies
Illegal name.  6RAGEDIES (ATTACH)
ER! a *>tragedies
OK, create hamlet
OK, a *>hamlet
OK, ld

<WRITER>SHAKESPEARE>PLAYS>TRAGEDIES>HAMLET (ALL access)
1 record in this directory, 1 total record out of quota of 0.

No entries selected.

OK, attach shakespeare
OK,
```

▶ AVAIL [disk-id] [–NORM]

AVAIL displays information on disk usage.

The information displayed for each disk consists of the diskname (listed after the label "Volume"), the total number of records used, the number of records available for use, and the percentage of records used. The default measurements are in

physical records (1 record = 2048 bytes), but can be given in normalized form with the –NORM option (1 normalized record = 880 bytes). In either case, the numbers are in decimal.

Use the STATUS DISKS command to list the names and logical device numbers of the disks connected to your system.

disk-id is one of the following:

Disk-id	*Meaning*
*	Displays information on all started disks.
–LDEV **n**	Displays information on the disk with the logical device number **n**, where **n** is a decimal number. To obtain the logical device number of a disk, use the STATUS DISKS command (to display the disk logical device numbers in octal) and then use the TYPE [OCTAL n] command to convert the octal number to decimal.
diskname	Displays information on the disk named **diskname**.

If you do not specify **disk-id**, information is displayed for the disk to which you are attached, provided that you have sufficient access rights. If you do not have these rights, the following message is returned:

```
Insufficient access rights
```

Examples of using AVAIL are shown below. Each example assumes that you are attached to a disk named MARKET, which has a logical disk number of 3 and to which you have sufficient access rights.

```
OK, AVAIL
      Volume    MARKET
       81477    total records
        4074    records available
       95.0%    full

OK, avail -ldev 3
      Volume    MARKET
       81477    total records
        4074    records available
       95.0%    full

OK, AVAIL MARKET
      Volume    MARKET
       81477    total records
        4074    records available
       95.0%    full

OK,
```

To check the status of all started disks, use AVAIL with an asterisk (*) as the argument. With this format, PRIMOS reads the file DISCS in the directory SYSTEM and prints a table of record utilization for all disks listed in the file. The columns under the COMMENTS heading list the information in the DISCS file, which is usually the logical and physical device numbers of each disk. The following example illustrates this format:

```
OK, avail *

Volume      Total      Free       %       Comments
ID          recs       recs     Full

- - - - - - - - - - - - - - - - - - - - - - - - - -

SHIP       140733      3924      97.2      0        4463
MARKET      59256      7080      88.1      1       32060
ACCNTS      37035     21503      41.9      2       71060
OK,
```

If the System Administrator has not set up the SYSTEM>DISCS file, the following error message is displayed when you issue the AVAIL * command:

```
NO DISCS FILE IN UFD SYSTEM
```

Similarly, if the System Administrator restricts access to disk information, all versions of AVAIL will fail.

See also STATUS.

▶ B

See BINARY.

▶ BACKUP pathname –MT n VOLID tapename [options] *Operator Command*

BACKUP produces full or incremental backups by copying file system objects from disk to tape.

Before using BACKUP, you must assign a tape drive with the ASSIGN command and mount the correct reel on the drive.

The command format for displaying an online help screen on BACKUP is

BACKUP –HELP [option]

For full details on the operation, arguments, and options of BACKUP, see the *Data Backup and Recovery Guide*.

▶ BACKUP_RELEASE –VOLID tapename [options]

Operator Command

BACKUP_RELEASE releases a BACKUP tape so that it can be overwritten.

The tape to be released must have been created with the BACKUP command. Before using BACKUP_RELEASE, you must assign a tape drive with the ASSIGN command and mount the correct reel on the drive. To display an online help screen on BACKUP_RELEASE, use the command format

BACKUP_RELEASE –HELP [option]

For full details on the operation, arguments, and options of BACKUP_RELEASE, see the *Data Backup and Recovery Guide*.

▶ BACKUP_RESTORE pathname [new-pathname] –MT n [options]

Operator Command

BACKUP_RESTORE restores file system objects from a BACKUP tape to disk.

Before using BACKUP_RESTORE, you must assign a tape drive with the ASSIGN command and mount the correct reel on the drive. The BACKUP tape must have been created with the BACKUP command.

To display an online help screen that lists the options and syntax of BACKUP_RESTORE, enter the command

BACKUP_RESTORE –HELP [option]

For full details on the operation, arguments, and options of BACKUP_RESTORE, see the *Data Backup and Recovery Guide*.

▶ BASIC [pathname]

BASIC loads the older Prime BASIC language interpreter.

When you invoke BASIC with a pathname, BASIC loads and runs the contents of **pathname** as a BASIC program and then returns to PRIMOS command level. Single-precision arithmetic is standard.

BASIC invoked without a pathname starts the BASIC interpreter, which displays a right angle-bracket (>) prompt and waits for a BASIC command. For further information on BASIC, refer to the *Interpretive BASIC Programmer's* Guide.

To invoke the newer BASIC language interpreter, use BASICV.

See also DBASIC; NUMBER.

▶ BASICV [–MIN] [pathname]

BASICV invokes the newer Prime BASIC language interpreter.

The BASICV interpreter is a virtual-memory subsystem, and is the recommended interpreter for executing BASIC programs. **pathname** is a file containing a BASIC program. Use the filename if the file is in your current directory. If you specify **pathname**, BASICV executes the program.

If you do not specify a pathname, BASICV enters Interactive mode and displays a right angle-bracket (>) prompt. You must then enter a BASICV subcommand. The following example illustrates the Interactive mode:

```
OK, basicv
BASICV REV19.4
>old bas.test
>list
bas.test       MON, SEP 24 1984          08:51:46


10 PRINT 'HELLO THERE'
20 PRINT 'SURE I AM WORKING'
30 END
>runnh
HELLO THERE
SURE I AM WORKING
>quit
OK,
```

The –MIN option checks that the program meets the American National Standard for MINIMAL BASIC.

The –PRX option forces the PRINT statement of BASICV to display a maximum of 14 digits of precision. Use it with or without the –MIN option.

For further information on BASICV, refer to the *BASIC/VM Programmer's Guide.*

See also NUMBER.

▶ BATCH option

BATCH, as a user command, displays information on active batch jobs. As an operator command, BATCH controls the processing of users' jobs.

BATCH as a User Command

The two user formats of the BATCH command are as follows:

```
BATCH -DISPLAY

BATCH -STATUS
```

Both options display information on jobs in the Batch subsystem.

The –DISPLAY Option

The –DISPLAY option provides detailed information in two tables. The first table lists the number of jobs waiting, deferred, and held in each queue. The second table lists the number of jobs currently executing and identifies each by user ID, job ID, phantom user number, and queue name.

Here is an example of the –DISPLAY option:

```
OK, BATCH -DISPLAY
[BATCH Rev. 20.0 Copyright (c) Prime Computer, Inc. 1985]

Number of waiting, deferred, and held jobs:

Queue       Jobs
- - - -     - - -
Normal-1    1
Normal-2    3

Total=    4 (2 queues)

2 currently running jobs:

  User        Jobid#     #     Queue
- - - -      - - - -    - -   - - - - -
PANCHO        #10023    114   Normal-2
CISCO         #00127    117   Normal-1
OK,
```

The –STATUS Option

The –STATUS option gives a one-line summary that includes the total number of waiting, deferred, and held jobs, the number of queues with waiting, deferred, and held jobs, and the number of executing jobs. The total number of active jobs is also given if there are waiting, deferred, and held jobs in addition to executing jobs. If there are no active jobs, the message `No batch jobs` is displayed.

Here is an example of the –STATUS option:

```
OK, BATCH -STATUS
[BATCH Rev. 20.0 Copyright (c) Prime Computer, Inc. 1985]
6 batch jobs; 4 waiting, deferred, or held jobs in 2 queues; 2 executing jobs.
OK,
```

BATCH as an Operator Command

BATCH accepts other options that are used only by the operator to control the processing of users' jobs. These options are –START, –STOP, –PAUSE, and –CONTINUE. The –START option can be given only from the supervisor terminal. The other three options can be issued from any terminal by the operator or by any user with ALL access to the BATCHQ directory.

For information on the operator options of BATCH, see the *Operator's Guide to System Commands* and the *Operator's Guide to the Batch Subsystem.*

▶ BATGEN option

BATGEN, as a user command, displays information on Batch queues. As an operator command, BATGEN controls the characteristics of Batch queues.

BATGEN as a User Command

The two user formats of the BATCH command are

BATGEN –STATUS

BATGEN –DISPLAY [queuename]

Both options display information on Batch queues.

The –STATUS Option

Use the following format to list the name and status for each Batch queue:

BATGEN –STATUS

The status of a queue can be blocked or unblocked, capped or uncapped, inactive, or flagged for deletion. An unblocked queue accepts jobs, whereas a queue that is blocked or flagged for deletion does not. A capped queue accepts jobs, but does not execute them until the System Administrator or the operator uncaps the queue. Jobs in an inactive queue are not executed until the queue enters its active window.

The following example illustrates the –STATUS option:

```
OK, BATGEN -STATUS
[BATGEN Rev. 20.0 Copyright (c) Prime Computer, Inc. 1985]

Queue:          Status:
- - - - - -     - - - - - - - - -
express         unblocked uncapped
normal-1        unblocked uncapped
normal-2          blocked   capped (inactive)
background-1    unblocked uncapped (inactive)
background-2    unblocked   capped

OK,
```

The –DISPLAY Option

Use the following format to display the name, status, and characteristics of Batch queues:

BATGEN –DISPLAY [queuename]

If you do not specify **queuename**, information is displayed for all queues. If you specify a queue name, information is displayed for only that queue.

The following example illustrates the –DISPLAY option:

```
OK, BATGEN -DISPLAY EXPRESS
[BATGEN Rev. 20.0 Copyright (c) Prime Computer, Inc. 1985]

Queue name = express, unblocked, uncapped.
Active window = FULL;
Default cptime=121, etime=6, priority=9;
Maximum cptime=120, etime=5; Funit=6;
Delta rlevel=0; Timeslice=99;

OK,
```

The first line of the output shows a queue's name and status (blocked or unblocked; capped or uncapped; inactive; or flagged for deletion). Unblocked and uncapped are a queue's default status.

The second line of the output shows a queue's **active window** — its time window of daily activity, within which its jobs are executed. The time window is specified in 24-hour format, hh:mm-hh:mm, or by the key word FULL if the queue is active at all times.

The last line of the output shows a queue's **delta rlevel**, that is, the number of levels a job's priority is lowered, at runtime, from the priority of the Batch monitor. A value of 0 (the minimum value) means a job runs at the same priority level as the monitor, while a greater value (7 is the highest numerical value) lowers the job

priority by that many levels. A value of IDLE (the maximum value and lowest priority level) means that a job runs only when the system is otherwise idle. **Timeslice** is the the amount of time (in tenths of seconds) a job execution receives before PRIMOS services the next user or process. See the JOB command for an explanation of the other parameters.

Note

If the BATDEF file is not read-enabled by the System Administrator, the BATGEN –STATUS and –DISPLAY commands produce error messages. In this case, users needing information about queues should consult the operator or the System Administrator.

BATGEN as an Operator Command

BATGEN accepts other options. Some are used by the System Administrator to configure, add, or delete queues. Other options are used by the operator to monitor, block, cap, unblock, and uncap queues and to set time windows on queues' daily activity. For further information on the operator's and administrator's use of BATCH, see the *Operator's Guide to the Batch Subsystem.*

See also BATCH; JOB.

▶ BINARY pathname

BINARY opens a file for writing on PRIMOS File Unit 3, usually as a binary output file for use by a compiler or assembler.

The file opened by BINARY is assigned the name **pathname**. If **pathname** is a filename, the file is opened in the current directory. This command has the same effect as the following OPEN command:

```
OPEN pathname 3 2
```

The CC, CBL, F77, FTN, PASCAL, PL1, PL1G, and VRPG compilers, and the PMA assembler, normally open a file named **basename**.BIN (if the source filename has the appropriate suffix) or B_**filename** (if the source filename has no suffix) as the binary output file. Use the BINARY command to send the output of several compilations to a single file.

▶ BIND [pathname] [options]

BIND creates an Executable Program Format (EPF) runfile from a binary file.

EPFs, which are dynamic V-mode or I-mode runfiles, are created by the BIND utility from binary files produced by the following compilers: CC, CBL, F77, FTN, PASCAL, PL1, PL1G, PMA, and VRPG. EPFs are executed with the RESUME command.

When issued without an option, BIND invokes the utility as an interactive program. It displays a colon (:) prompt and waits for a BIND subcommand

```
OK, BIND
[BIND Rev. 21.0 Copyright (c) 1985, Prime Computer, Inc.]
:
```

Enter one BIND subcommand per line. Enter the HELP subcommand to display BIND's online help facility.

When you issue BIND with the appropriate options, the entire linking session is performed from the command line. With few exceptions, you can specify a BIND subcommand as a command line option if you precede it with a hyphen. For example, LIBRARY is a subcommand and –LIBRARY is a command line option.

For details on EPFs and on the operation and options of BIND, see the *Programmer's Guide to BIND and EPFs*.

▶ BOOT_CREATE [pathname] [options]

Operator Command

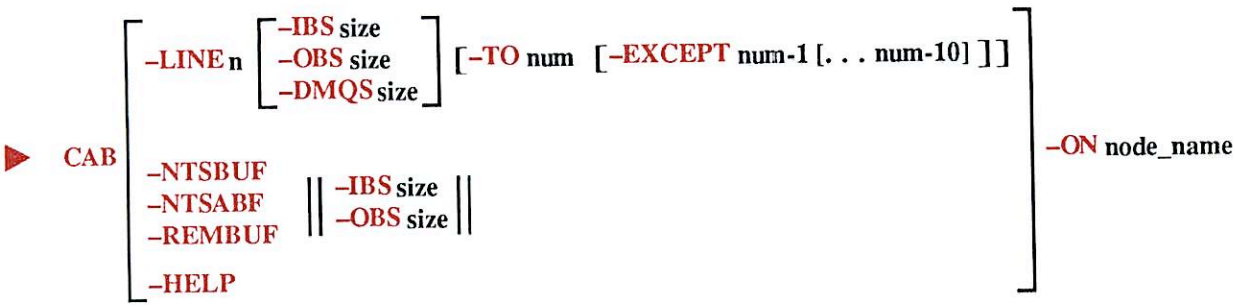
BOOT_CREATE creates a boot tape suitable for booting the system from magnetic tape.

BOOT_CREATE opens and reads a file (specified by **pathname**) that lists the pathnames of the files and directories to be saved on the boot tape. If you do not supply **pathname**, you are prompted for it. Enter BOOT_CREATE –HELP to display an online help facility that lists the command's syntax and options.

For further details on BOOT_CREATE, see the *Operator's Guide to System Commands* and the *Data Backup and Recovery Guide*.

▶ C

See CLOSE.



CAB changes the sizes of a line's asynchronous buffers interactively or at cold start. Use this command interactively at the supervisor terminal or include it in the system PRIMOS.COMI file.

This command is used by the system operator, the DSM Administrator, and DSM-privileged users.

For more information, see *System Administrator's Guide, Volume II: Communication Lines and Controllers*.

See also LAB.

▶ CBL pathname [options]

CBL invokes the COBOL 74 compiler.

pathname is the name of the source file to be compiled. Use a filename if the file is in your current directory. It is recommended that you give **pathname** a .CBL suffix.

For detailed information on the operation and options of CBL, see the *COBOL 74 Reference Guide*.

▶ CBLDML in-pathname [[–OUTPUT] out-pathname] [[–ERROR] err-pathname] [–DYNAMIC] [–NO_LINE_NUMBER]

CBLDML invokes the COBOL 74 Data Manipulation Language (DML) preprocessor.

The DML preprocessor translates embedded DML statements in the source file specified by **in-pathname** into a form that is usable by the COBOL 74 compiler. The preprocessor automatically adds a .CBL suffix to the filename part of **in-pathname**. **out-pathname** is the name of the output file that contains the translated DML statements. **err-pathname** is the file that contains the error listing.

The output file is placed in the current directory, unless you use the –OUTPUT option to place it in another directory.

Use the –ERROR option to place the error listing in a directory other than your current one.

–DYNAMIC allows programsto invoke a schema at runtime by a schema name rather than by the schema number obtained at compile time. The schema name is used at runtime to dynamically resolve the schema number.

–NO_LINE_NUMBER suppresses the generation of line numbers in the output file.

For further information, see the *DBMS Data Manipulation Language Reference Guide*.

▶ CBLSUBS [in-pathname] [–OUTPUT out-pathname] [–LIST list-pathname]

CBLSUBS invokes the COBOL 74 Subschema Data Definition Language (DDL) compiler.

in-pathname is a source file containing the subschema definition. The CBL Subschema DDL compiler translates **in-pathname** into the subschema table (**out-pathname**). The subschema table is placed in the same directory as the schema table. To place the subschema table in another directory, specify the –OUTPUT option.

The compiler also produces a listing file that contains the source listing, error messages, and a map of the User Work Area. The listing file is named **in-pathname**.LIST and is placed in the directory that contains the schema table. Use the –LIST option to specify another directory for the listing file or a different name for the file.

If you enter CBLSUBS without arguments, you are prompted for the name of the source file:

```
OK, CBLSUBS
Source name?
:
```

For more information on CBLSUBS, see the *DBMS Data Description Language Reference Guide*.

▶ CC pathname [options]

CC invokes the Prime C compiler and produces an object program from an ASCII source file named **pathname**.

It is recommended that you give **pathname** a .C or .CC suffix. For details on the operation and options of the C compiler, see the *C User's Guide*.

▶ CHANGE_PASSWORD { old-password | -PROMPT }

CHANGE_PASSWORD changes a user's login password.

A password is a string of as many as 16 characters, known only to you. The string may contain any ASCII characters, with the exception of the following PRIMOS reserved characters:

( ) { } [ ] < > ~ ‘ ! @ % ^ + = ’ " , ; : ? | \
space rubout/delete

After you enter the old password, the system prompts you twice for the new password. For security reasons, the new password is not echoed at the terminal. Your password is changed only if you enter the old password correctly and if both new passwords match.

The –PROMPT option makes the system prompt you for your old password so that you do not have to enter it on the command line. When you type the old password, it is not echoed on the screen. The rest of the procedure is the same as without this option.

If you do not have a login password (that is, if you have a null password), you can create a password by typing CHANGE_PASSWORD followed by a carriage return, and then entering the new password in response to the system prompts.

Errors While Changing Passwords

The list below describes errors you may commit while changing your password:

- You enter the old password incorrectly.
- You enter the second new password differently from the first during the confirmation dialogue. Both new passwords must match.
- You enter a new password that is shorter than the minimum length allowed by your system. Your System Administrator determines the minimum length of a password.
- You attempt to create a new password that is the same as the old password.
- You attempt to create a null password on a system that does not allow them. Your System Administrator determines if null passwords are allowed.

These errors cause the CHANGE_PASSWORD command to fail and produce appropriate error messages.

If any of these errors occur, your old password remains in effect.

Examples of Changing Passwords

The next three examples illustrate the CHANGE_PASSWORD command.

Example 1: The following example illustrates CHANGE_PASSWORD without the –PROMPT option. You must enter the old password (ROSEBUD) on the command line. The new password is not echoed at the terminal:

```
OK, CHANGE_PASSWORD ROSEBUD
New password:    Enter a new password.
Reenter new password for confirmation:  Enter new password again.
OK,
```

Example 2: In the following example, the use of the –PROMPT option means that you do not enter the old password on the command line, but rather at the first CHANGE_PASSWORD prompt. Neither the old password nor the new password is echoed at the terminal during the dialogue:

```
OK, CPW -PROMPT
Old password:  Enter your old password.
New password:  Enter the new password.
Reenter new password for confirmation:  Enter new password again.
OK,
```

Example 3: The following example illustrates the error message produced by a mismatch of new passwords:

```
OK, change_password baron
New password?  Enter the new password MARS.
Reenter new password for confirmation: You enter MARX instead of
MARS.
Passwords do not match!  (change_password)
ER!
```

Note

If you forget your current password, you must contact the System Administrator because only the System Administrator can create a new password for you.

▶ CHAP argument

CHAP, as a user command, changes a user's execution priority level. As an operator command, CHAP changes the priority level and timeslice of one or all users.

The operator assigns each user or process (including phantom processes) on a Prime system a range of execution priority levels. The general-purpose levels range from 0 (lowest priority) through 3 (highest priority). (If no range is assigned, the default is 1 for the upper level and 0 for the lower level.) The higher your priority level, the more attention you receive from PRIMOS.

There are two special-purpose priority levels, IDLE and SUSPEND. A user or process assigned an IDLE level is serviced by PRIMOS only when no other users or processes at any other priority are eligible to run (that is, when the system is idle). Users can assign an IDLE level to their phantom processes but not to themselves. An IDLE process retains that level until it is changed with CHAP to another priority level.

A user or process assigned a SUSPEND level cannot be serviced by PRIMOS until the operator changes the level with CHAP to another priority level. Only the operator can suspend the priority level of a user or process.

CHAP as a User Command

The user form of CHAP allows you to change your execution priority level within the range assigned by the operator. For example, if your priority level is 2, you can vary the level between 0 and 2. If your priority level is 1, you can lower it to 0 and then raise it again to 1, but you cannot raise it above 1.

The user format of CHAP is

```
CHAP { DEFAULT
       DOWN
       IDLE
       LOWER n [tenths]
       -TIMESLICE milliseconds
       UP }
```

The meanings of the arguments are

Argument	*Meaning*
DEFAULT	Resets your priority level to your current default level. Especially useful to return a process from the IDLE priority level.
DOWN	Lowers your priority level by one level (but no lower than your lowest assigned level). If you are already at your lowest level (usually 0), the command does nothing. You cannot use DOWN to move a process into the IDLE or SUSPEND level. CHAP DOWN is equivalent to the CHAP LOWER 1 command.
IDLE	Places a user's process at the IDLE priority level. You can issue this command format only from a phantom process, not from your terminal. Use CHAP DEFAULT to return to your default priority level.

LOWER n [tenths]	Lowers your priority level by **n** levels (but no lower than your lowest assigned level). **n** is an integer from 0 to 7, inclusive. The optional **tenths** argument assigns you a new timeslice value. (Your timeslice value determines the amount of time that PRIMOS gives to your process before it goes on to the next user.) **tenths** is an octal integer that specifies tenths of a second (for example, octal 24 is decimal 20-tenths of a second, which is two seconds.) The value of **tenths** must be less than your current timeslice value (that is, you can only shorten your timeslice value, not lengthen it). You cannot use LOWER to move a process into the IDLE or SUSPEND level.
–TIMESLICE milliseconds	Specifies your new timeslice value in milliseconds (thousandths of a second). **milliseconds** must be a decimal number (not octal, as with the tenths argument above). If **milliseconds** is 0 or missing, your timeslice is reset to the default value for your machine; if **milliseconds** is -1, your timeslice is effectively infinite. Otherwise, **milliseconds** must be between 4 and 32767.
UP	Raises your priority level by one (but no higher than your highest assigned level). CHAP UP does nothing if you are already at your upper level or if your process is at the IDLE priority level.

CHAP as an Operator Command

CHAP is used by the operator to assign an execution priority level and, optionally, to change the timeslice (the amount of time that a user has for processing) for a specified user or for all users. The operator format of CHAP is

```
CHAP { -userno } { -IDLE                          }
     { ALL    } { -SUSPEND                        }
               { priority [tenths]                }
               { -TIMESLICE milliseconds          }
```

For details on the operator form of CHAP, see the *Operator's Guide to System Commands*.

▶ **CLOSE** { **pathname** | **–UNIT funit-1 [...funit-16]** | **–ALL** }

CLOSE closes one or more files.

The CLOSE command closes files in one of three ways: by pathname, by file unit number (**funit**), or by closing all files (except a command output file) at once. In all cases, issuing CLOSE for a file that is already closed does not result in an error.

Closing a File by Name

To close a file by specifying its name, use the format

CLOSE pathname

pathname is the name of the file. If **pathname** cannot be found, an error message is displayed and you are returned to PRIMOS command level.

When this format is issued from the supervisor terminal, it closes access to the specified file for all users.

Closing a File by a File Unit Number

To close one or more files by specifying file unit numbers, use the format

CLOSE –UNIT funit-1 [...funit-16]

funit is a decimal number specifying any legal file unit number. (File unit numbers range from 1 to 32,762, depending on the configuration of the system.) You can specify a maximum of 16 file units.

Closing All Files at Once

To close all your files at once, use the following format:

CLOSE –ALL

This format closes all open files except for a command output file. You must close command output files by pathname or by using the COMOUTPUT –END command.

In a command file or CPL program file, specify each item to be closed. If you use CLOSE –ALL from within either file, the file itself will be closed and its execution terminated.

CLOSE –ALL ensures that buffers are retrieved properly and that the state of the file system is reset. CLOSE –ALL is particularly useful when you are uncertain as to the state of the files in the current directory, because you can issue CLOSE –ALL even if all your files are closed.

If you stop a program by pressing the BREAK key or issuing the CONTROL-P key sequence, you should issue a CLOSE –ALL command. Otherwise an error message may result when you enter a subsequent command. After CLOSE –ALL has been given, the stopped program cannot be continued (started).

Older Form of CLOSE

An older form of the CLOSE command uses the syntax

CLOSE { **pathname** / **funit-1 [...funit-n]** / **ALL** }

funit is an octal number of the file unit you want closed. The other two arguments work like their counterparts in the newer form of CLOSE.

See also OPEN.

▶ CLUP [–USERNO user-number]

CLUP invokes the ROAM Command Processor Clean-up Program.

To perform the cleanup procedure at your terminal, use CLUP without the –USERNO option. This option is used to perform a remote cleanup for a phantom user, a batch job, or a user who has logged out. **user-number** is the user's terminal number or phantom number. For details, see the *ROAM Administrator's Guide* or the *DBMS Administrator's Guide*.

▶ CMPF file-a file-b [file-c...file-e] [options]

CMPF compares lines in a maximum of five ASCII files and shows textual differences.

The files to be compared can be specified by pathnames or by filenames if the files are in your current directory. **file-a** is the original file against which the other files are compared and is treated as the common ancestor of those files. **file-b** through **file-e** are the files to be compared.

The CMPF output lists text lines in the new files that were added, changed, or deleted from **file-a**. The CMPF command, along with the MRGF command, is particularly useful for parallel software development.

The CMPF options are

Option	*Meaning*
–BRIEF	Displays only the file identification letter and the line number of a line that differs, but not the text itself.
–MINL number	Sets the minimum number of lines that must match (after a difference in the files being compared is found) in order to resynchronize file comparison. (Default is 3.)
–REPORT pathname	Creates a file named **pathname** and directs the CMPF output to it instead of to the terminal.
–STOP	Stops the comparison process as soon as one difference among the files is discovered. If a difference exists, displays a **FILES DO NOT MATCH** message.

CMPF Operation

When a difference is found between **file-a** and any other file, CMPF attempts to resynchronize the files for further comparison. Resynchronization is accomplished only when a certain minimum number of lines (specified in –MINL) match in all the files being compared. After resynchronization, lines that differ from **file-a** are displayed at the terminal or written to the report file. The comparison process then continues until another difference is found.

When line differences are reported, CMPF identifies each line from **file-a** with the letter A and a line number. The corresponding lines of the other files are indicated in the same manner, using letters B through E, respectively.

Example of CMPF

For this example, two files, named FOX_ONE and FOX_TWO, are used. FOX_ONE is **file-a**, the original file.

```
FOX_ONE          FOX_TWO
The              The
quick            swift
brown            red
fox              fox
jumps            jumps
over             over
the              the
lazy             the
dog              dog
```

A CMPF comparison of these two files works as follows:

```
OK, CMPF FOX_ONE FOX_TWO
[CMPF 21.0]

A2      quick
A3      brown
CHANGED TO
B2      swift
B3      red

A8      lazy
DELETED BEFORE
B8      dog

COMPARISON FINISHED.
2 DISCREPANCIES FOUND.

OK,
```

Note

CMPF compares compressed lines of any length and assumes that files of common ancestry contain lines compressed in an identical fashion. It is therefore possible for a mismatch to occur between two lines that appear identical but were compressed differently.

See also FILVER; MRGF.

▶ CNAME old-name new-name

CNAME changes the name of a file system object. The file system object must be closed before you can change its name.

old-name is the name of the file, directory, segment directory, or access directory you want changed. **old-name** can be specified by a pathname but **new-name** cannot. If you specify a pathname for **old-name**, **new-name** replaces only the final element of the pathname.

To use CNAME, you must have Delete (D) and Add (A) rights for an ACL directory or owner status for a password directory. You cannot change the name of a directory while attached to that directory. However, you can change a directory name while attached to one of its subdirectories.

Example 1: The following example changes the name of a file from STUDENTS.PASCAL to CLASS.PASCAL:

```
OK, CNAME STUDENTS.PASCAL CLASS.PASCAL
OK,
```

Example 2: The following command changes the name of the directory JONES in the partition ADMIN to WILSON:

```
OK, CNAME <ADMIN>JONES WILSON
OK,
```

Note

Do not change the names of special directories such as CMDNC0.

▶ CN_RBF old-filename new-filename [–ALL]

CN_RBF changes the name of an active ROAM master or slave file.

The new name assigned by CN_RBF appears in the user file directory and in the system recovery table (RCVTAB) file. For DBMS files, this command changes the name of the entire segment directory as well as the schema name.

old-filename, which can be a pathname, is the existing name of the file you want changed. **new-filename** specifies the new name of the file and must be a filename (entryname), not a pathname.

The –ALL option changes the names of the master segment directory and of all slave segment directories to **new-filename**. This option is the default for DBMS files.

For further information on CN_RBF, see the *ROAM Administrator's Guide*.

▶ CO

See COMINPUT.

▶ COMINPUT { pathname [funit] / option }

COMINPUT executes a command input file or controls its command flow.

A command input file (also called a COMI file) contains a series of commands, each on a separate line. These commands can be any legal PRIMOS command, utility subcommand, or dialog response. When you execute the command input file, PRIMOS takes its commands from the file instead of from your terminal. The commands are processed as if you had entered them at your terminal.

You construct command input files with the ED and EMACS text editors. Command input files are especially useful for repetitive processes such as compiling and loading a series of programs, building libraries, and running production jobs.

Executing a Command Input File

Use the following command format to execute a command input file:

COMINPUT pathname [funit]

pathname specifies the command input file from which input is to be read by PRIMOS. If the file is in your current directory, you can specify a filename instead of a pathname.

funit specifies the PRIMOS file unit number, in octal, on which **pathname** is opened. If you omit **funit**, File Unit 6 is the default.

COMINPUT Options

After PRIMOS begins to process a command input file, you can control the file's command flow with the COMINPUT options listed below. You can use only one option at a time.

The –END, –TTY, and –PAUSE options should be used from within a command input file, not from your terminal. Conversely, you should use the –START option only at your terminal. You can use the –CONTINUE option from your terminal or from within the file.

The COMINPUT options are

Option	*Meaning*
–CONTINUE [funit]	Resumes reading commands from a command input file after a COMINPUT –PAUSE command or an error. **funit**, which is an octal number, must be specified if the file is opened on a file unit other than the default File Unit 6.
–END	Closes the command input file and returns to the terminal for input.
–PAUSE	Returns to the terminal for input but does not close the command file.
–START [funit]	Restarts processing of a command file after BREAK, CONTROL-P, a warm start of PRIMOS, or a subsystem error. **funit**, which is an octal number, must be specified if the file is opened on a file unit other than the default File Unit 6.
–TTY	Closes the command input file and returns to the terminal for input. (Same as –END.)

Using COMINPUT

Here are some guidelines for using COMINPUT:

- If you do not specify a pathname or an option when issuing COMINPUT, the following error message is displayed:

```
OK, COMINPUT
Pathname or option must be specified. (COMINPUT)
ER!
```

- After a COMINPUT –PAUSE command, you can invoke other commands, or use the following form to start another command file on another unit:

  COMINPUT pathname funit

- To resume processing the original command file, use the command format

  COMINPUT –CONTINUE funit

- Do not use the –START option if you entered PRIMOS command level from a command file process by means other than a BREAK or CONTROL-P, such as by an ER! error message. In these instances, use the following command format to continue processing a command file:

  COMINPUT –CONTINUE [funit]

  You should use COMINPUT –START if the quit is from inside a processing program, and COMINPUT –CONTINUE if an error in a command line in the command file causes an automatic COMINPUT –PAUSE.

- Any error message that occurs during the processing of a command file causes the command input stream to be switched to the terminal. However, the command input file remains open, which allows you to reinvoke the command that caused the error message and, subsequently, to resume the command file at the command following the one that caused the error.

- You can insert comments in a command input file by using the format

  /* text

  The slash and asterisk characters (/*) begin the comment, which ends at the end of the line. A comment may also be appended to a command line within the file, as in the following example:

```
SLIST BENCH.MAP /*PRINT MAP FILE
```

  Comments may also be inserted at subcommand level for certain utilities such as the SEG LOAD subprocessor.

- You can chain command files with the –CONTINUE option. The last command file in the chain must be terminated with a COMINPUT –TTY or COMINPUT –END to ensure that files opened in the process are closed and that the chain is terminated properly. You must keep track of which file units are being opened and closed as the chain progresses, especially those file units that are opened for command input files.

- Do not use the command CLOSE –ALL within a command file because it closes the command file with the following error message:

```
End of file. Cominput. (Input from terminal.)
```

For further information on COMINPUT and command input files, see the *PRIMOS User's Guide*.

See also CPL.

▶ COMM_CONTROLLER main_subcommand [option]

Operator Command

COMM_CONTROLLER enables you to shut down or reload an ICS1, ICS2, ICS3, LHC300, or LTS300 controller without having to perform a cold start. COMM_CONTROLLER also lets you upline dump the memory of the LHC controller to a disk file and verify the integrity of an ICS controller.

For more information about COMM_CONTROLLER, and for further information on controllers, downline loading, and upline dumping, see the *ICS User's Guide*.

▶ COMOUTPUT { pathname [options] / options }

COMOUTPUT opens a command output file and sends to it the output produced by the user at the terminal or by a command file.

A command output file (also called a COMO file) can store everything that you see on your terminal. This includes all commands that you (or a command file) enter and all output produced by those commands.

You can therefore use a command output file as a record of your terminal transactions. After you close the file, you can display its contents at your terminal with the SLIST command or print it with the SPOOL command.

Opening a Command Output File

To open a command output file, use the format

COMOUTPUT pathname [options]

pathname is the name of the command output file. You can use a filename instead of a pathname to place the file in your current directory.

The file is opened on a special file unit called COMO. (The actual file unit number is -4.) In addition to opening the file, the command also begins sending all PRIMOS output to the file, as well as to your terminal. To send the output only to the file, use the –NTTY option.

If **pathname** does not exist, the command creates it. If **pathname** already exists, the file is overwritten, unless you specify the –CONTINUE option. The –CONTINUE option appends input to the end of the file.

Use the –PAUSE option (with or without –CONTINUE) if you want to open a command output file but do not want to send output to it immediately. Issue a subsequent COMOUTPUT –CONTINUE command to send output to the file.

You can have only one command output file open at any given time. If you open a second command output file before closing the first one, the second COMOUPUT command automatically closes the first file before it opens the second.

Use the STATUS UNITS command to find out if you have a command output file opened, as shown below:

```
OK, COMO DATA.COMO
OK, STATUS UNITS


User GARRY                                    SYS6

File     File       Open     File
Unit     Position   Mode     Type   RWlock   Treename
COMO     000000030     W      DAM    NR-1W   <TRANS>GARRY>DATA.COMO


OK,
```

Closing a Command Output File

Use the following command to close a command output file:

COMOUTPUT –END

You can also use the CLOSE command if you specify the filename.

COMOUTPUT Options

The COMOUTPUT options are described below. You can specify two or more options if they do not conflict.

Option	*Meaning*
–CONTINUE	Appends the PRIMOS output to a command output file. If the file is not open **pathname** must precede this option. If the file is open but was suspended with –PAUSE, this option does not take a pathname because it starts sending PRIMOS output again to the suspended file.
–END	Stops sending command output to a file and closes it. This is the recommended method of closing a COMOUTPUT file.
–NTTY	Turns off output to the terminal (that is, does not display the `OK,` prompt or responses to command lines), but does not close the command output file. Terminal output is resumed when one of the following occurs: you press the BREAK or CONTROL-P keys, an error occurs, or –TTY is specified in a subsequent COMOUTPUT command.
–PAUSE	Suspends command output to the command output file, but does not close the file. You must subsequently issue a COMOUTPUT –CONTINUE or –END command.
–TTY	Turns on output to the terminal but does not close the command output file. Used after a COMOUTPUT –NTTY command.

Example of Using COMOUTPUT

The following series of COMOUTPUT commands illustrates the effects of the options:

1. The following command opens a command output file named REPORT and writes to it any subsequent terminal output:

```
OK, COMO REPORT
```

   Commands and echoed responses are also displayed at the terminal. REPORT is overwritten if it already exists.

2. The following command stops writing command output to REPORT, but does not close it:

```
OK, COMO -PAUSE
```

   Output continues to be displayed at the terminal.

3. To resume writing command output to REPORT, use the following command:

```
OK, COMO -CONTINUE
```

4. The following command continues to send terminal output to REPORT, but the output does not appear on the terminal screen because the –NTTY option turns off terminal display:

```
OK, COMO -NTTY
```

5. Use the following command to close the REPORT file:

```
COMO -END
```

The **OK,** prompt is not displayed because of the previous –NTTY option.

6. To turn on terminal display, use the following command:

```
COMO -TTY
```

7. The following command reopens the file named REPORT and positions it to end-of-file, but does not send terminal output to the file because of the –PAUSE option:

```
OK, COMO REPORT -C -P
```

Use the –CONTINUE option when you want to append output to an existing file.

8. The following command resumes writing terminal output to REPORT:

```
OK, COMO -C
```

9. Close the file with the following command:

```
OK, COMO -END
```

For further information on COMOUTPUT and command output files, see the *PRIMOS User's Guide*.

▶ CONCAT [out-pathname] [options]

CONCAT concatenates a number of input files into one output file suitable for spooling.

out-pathname is the name of the output file. Use a simple filename to place the file in your current directory.

You can omit **out-pathname** if you previously opened a file (with the OPEN command). File Unit 2 is the default output file unit, but you can specify another output file unit with the –OUNIT option.

If **out-pathname** is omitted and no file is open, CONCAT aborts and displays the following error message:

```
Output file not open (CONCAT)
ER!
```

CONCAT Modes

CONCAT begins its process in either Insert mode (which is the default mode) or in Command mode (if you specify the –COMMAND option on the command line). A colon prompt (:) means that CONCAT is in Insert mode; a right angle-bracket prompt (>) identifies Command mode.

Insert Mode

If you do not specify the –COMMAND option on the command line, CONCAT enters Insert mode. The next step depends on whether the output file is a new file or an existing one.

Using a New Output File: If **out-pathname** is a new file, CONCAT displays the colon prompt so that you can enter the names of the input files

```
Enter filenames, one per line:
:
```

Enter only one filename (in either lowercase or uppercase) at each colon prompt. To finish specifying filenames, enter a blank line (that is, press RETURN without typing anything). A blank line changes CONCAT to Command mode, where you can enter any CONCAT subcommand (including INSERT to return to Insert mode or QUIT to return to PRIMOS command level).

The following example illustrates a simple CONCAT operation. OUT_FILE is the name of the output file:

```
Enter filenames, one per line:
: ALPHA
: BETA
: OMEGA
:

> QUIT
OK,
```

The input files ALPHA, BETA, and OMEGA are now concatenated into the output file OUT_FILE. The original input files, however, have not been altered.

Using an Existing Output File: If **out-pathname** is an old file and you do not specify either the –APPEND or the –OVERWRITE option, CONCAT first asks you how to modify the file and then requests the names of the input files.

The following example illustrates this process, using RESULT as the name of the output file.

```
OK TO MODIFY OLD RESULT? yes
OVERWRITE OR APPEND: a

Enter filenames, one per line:
:
```

Answering N or NO to the first prompt aborts CONCAT and returns you to PRIMOS command level. Answering Y, YES, O, or OK continues the CONCAT procedure.

At the second prompt (**OVERWRITE OR APPEND**), either answer OVERWRITE (or O) to replace the old file with the new one, or answer APPEND (or A) to preserve the contents of the old file and add the new input files at the end of the old file.

All answers can be entered either in lowercase or uppercase. The rest of the CONCAT operation is the same as described above.

Command Mode

CONCAT enters Command mode if you specify the –COMMAND option on the command line or if you enter a blank line at the Insert mode colon (:) prompt.

The Command mode prompt is a right angle-bracket (>). At this prompt, you can enter any CONCAT subcommand listed below. (Any subcommand, except QUIT, used after the INSERT subcommand has no effect on the output file.) Enter one subcommand per line. Blank lines are ignored, causing another right angle-bracket prompt to be displayed. Any text preceded by a slash and an asterisk (/*) is taken to be a comment and is also ignored.

The INSERT subcommand inserts the desired files but does not cause CONCAT to leave Command mode. To exit Command mode, issue the QUIT subcommand. QUIT ends CONCAT and returns you to PRIMOS command level.

Options and Subcommands

CONCAT instructions are either options or subcommands. Options can be given only on the command line and must be preceded by a hyphen. Subcommands can be given only in Command mode and cannot be preceded by a hyphen.

Many options can be given as subcommands if the initial hyphen is omitted. For example, –HEADER is a command line option, whereas HEADER is a subcommand.

Command-line Options

The options below can be given only on the command line. You may enter them in any order.

Option	*Meaning*
–APPEND	Preserves the contents of **out-pathname** (which is an existing file) and appends the input files to its end.
–CLOSE	Truncates and closes the output file on exit. (Default)
–COMMAND	Begins CONCAT in Command mode with the right angle-bracket (>) prompt.
–INSERT	Begins CONCAT in Insert mode with the colon (:) prompt. (Default)
–IUNIT n	Specifies the file unit on which an input file is open. (Default input unit is 1.)
–OPEN	Leaves the output file open on exit but does not truncate it.
–OUNIT n	Specifies the file unit on which the output file is open. (File Unit 2 is the default.) If **out-pathname** is omitted from the command line, the file open on unit **n** is used for output.
–OVERWRITE	Writes over **out-pathname** (which is an existing file), erasing its contents.
–TRUNCATE	Truncates the output file on exit but leaves it open.
–VERIFY	If **out-pathname** already exists, asks you (with the `OK TO MODIFY OLD` prompt) if the file should be modified. If you answer YES or OK, then asks you (with the `OVERWRITE OR APPEND` prompt) how to modify the file. (Default)

Options/Subcommands

The following instructions may be given as options on the command line (if they are preceded by a hyphen) or as subcommands in Command mode (without a hyphen). If specified, the –BANNER option must be the last option on the command line.

Option/Subcommand	*Meaning*
BANNER [banner-line]	Generates both titles and banner pages. Banner pages are inserted between input files. A banner page consists of two lines, each containing as many as 14 large characters. **banner-line** specifies the first line and is read as raw text. (Spaces are therefore accepted.) If you omit **banner-line** the first line will be blank. The second line is the entryname (last component) of the input file pathname. Titles are generated as in HEADER.
DELETE	Deletes input file after copying it to the output file. This option has no abbreviation.
EJECT	Generates a page eject between input files and suppresses both titles and banner pages.

HEADER	Generates titles but not banner pages. (Default) If the first line of an input file is a title (that is, begins with octal 1 in the left byte), then the line appears only as the title for the file. Otherwise, the line appears both as the title line and as the first line of the file.
NDELETE	Does not delete input file after copying it. (Default)
NHEADER	Suppresses both titles and banner pages. The input files are copied to the output file without modification.
NRESETP	Does not reset spooler page numbering between input files. (Default)
RESETP	Resets spooler page numbering between input files.

Subcommands

The following three subcommands can be given only in Command mode and not on the command line.

Subcommand	*Meaning*
INSERT [file-list]	If you omit **file-list**, CONCAT enters Insert mode to accept the names of the files to be concatenated. Enter one filename per line. To exit from Insert mode, press RETURN without typing anything. If you specify a list of files, the files are concatenated into the output file without entering Insert mode. A maximum of 40 files may be specified on one line, separated by spaces or commas. Pathnames with embedded spaces (that is, passwords) must be enclosed in quotes. If an error exists in the line, the rest of the line after the error is ignored.
QUIT	Exits from CONCAT. This is the only way to exit from CONCAT without leaving one or more file units open.
TITLE [new-title]	Uses **new-title** as the banner page for the next input file. **new-title** is read as raw text, so that spaces are accepted. If **new-title** is omitted, the filename is used.

▶ CONFIG –DATA config-pathname

Operator Command

CONFIG is used by the System Administrator to define system parameters when the system is cold started. **config-pathname** is the name of the file that contains the system configuration.

This is the only currently supported form of the CONFIG command.

For detailed information, see the *System Administrator's Guide, Volume I: System Configuration.*

▶ CONFIG_DSM [options]

Operator Command

CONFIG_DSM invokes a menu utility that enables you to create and modify a Distributed System Management (DSM) configuration file. The command line options allow you to set the terminal environment so that CONFIG_DSM works properly.

DSM provides networked systems management from any convenient point on the network. For further information, see the *DSM User's Guide*.

▶ CONFIG_NET [{ [pathname] [–TTP terminal-type] / –HELP }]

Operator Command

CONFIG_NET allows a Network Administrator to create or modify a global configuration file that is used to configure a PRIMENET™ network.

pathname is the name of the file that contains an existing network configuration. The –TTP option specifies your terminal type (default is PST 100). The –HELP option displays the command's syntax and values for –TTP. For detailed information, see the *PRIMENET Planning and Configuration Guide*.

▶ CONFIG_NTS [config_pathname] [options]

Operator Command

CONFIG_NTS invokes the Network Terminal Support configuration program.

The CONFIG_NTS command allows the network operator to create, edit, or list an NTS configuration file. CONFIG_NTS is part of the Network Terminal Service (NTS), which allows remote LTS terminals to establish a connection with a Prime host system, and also manages local asynchronous services. For detailed information, see the *NTS Planning and Configuration Guide*.

▶ CONFIG_UM selection [options]

Operator Command

CONFIG_UM controls event message handling on either local or remote nodes by configuring the Distributed System Management (DSM) facility's Unsolicited Message Handler (UMH) through user selections. User access to CONFIG_UM is controlled by the System Administrator.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ CONVERT_AMLC_COMMANDS

Operator Command

CONVERT_AMLC_COMMANDS allows the user to process AMLC commands from a terminal or from a file and convert these commands into the appropriate form of the SET_ASYNC command. For more information, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

▶ CONVERT_ENV

Operator Command

CONVERT_ENV invokes a utility that converts pre-Rev. 21.0 spooler environment files to Rev. 21.0 spooler environment files. Only the administrator of the Spooler subsystem may use this utility. For more information, see the *Operator's Guide to the Spooler Subsystem.*

▶ CONVERT_TO_ACLS { –DEFAULT | –NONE }

Operator Command

CONVERT_TO_ACLS invokes a utility that converts all password directories on a partition to Access Control List (ACL) directories. The user must have rights to set a priority ACL before using CONVERT_TO_ACLS. For further information, see the *System Administrator's Guide, Volume III: System Access and Security.*

▶ COPY source-pathname [target-pathname] [options]

COPY copies file system objects either from one directory to another or within a directory.

source-pathname is the pathname of the object to be copied (source object). **target-pathname** is the pathname to be given to the copied object (target object). See the sections below for details on specifying the source and target pathnames and on the COPY options.

The copy operation does not alter the source object in any way, nor does it remove it from its original directory, unless you specify the –DELETE option.

Required Access Rights

To use COPY, you must have the following rights:

- Read (R) access to both the source directory (the directory that contains the object you want copied) and to the source object.
- Add (A) access to the target directory (the directory that will contain the copied object).

- Delete (D) access to the target directory if an object with the same name as the target object (the copied object) exists. You must also have Delete access to the source directory to use the –DELETE option.
- Protect (P) access if you are working with an ACL directory and use the –COPY_ALL, –DTM, –PROTECT, –QUOTA, or –RWLOCK options.

Specifying a Source Pathname

source-pathname is the pathname of the object to be copied (source object). The source object can be a file (including an EPF file), directory, segment directory, or access category.

If the source object is in your current directory, you need only specify its entryname. For example, to copy the file DAISY from your current directory FLOWER to the directory GATSBY, you issue the command

```
OK, COPY DAISY GATSBY>DAISY
```

Your file FLOWER>DAISY now also exists as GATSBY>DAISY.

Specifying a Directory or Access Category

If **source-pathname** is a directory or access category, PRIMOS asks for your confirmation before copying it, as in the following example:

```
OK, copy <bird>sparrows <fish>trout>rainbow
Ok to copy directory "<BIRD>SPARROWS" to "<FISH>TROUT>RAINBOW"? yes
OK,
```

Use the –NO_QUERY option to copy the directory or access category without the verification query.

When copying a directory tree, you can use the –LEVELS option to copy only a specified number of levels in the tree. For example, the command below copies the top three levels of the directory tree ORCHARD to the directory FOREST:

```
COPY ORCHARD FOREST -LEVELS 3
```

See the section below, Copying With Password Directories, for details on using COPY with password directories.

Specifying a Segment Directory

COPY allows the use of segment directory file offset numbers in the source or target pathname. PRIMOS does not query you for verification before proceeding with the copy operation:

```
OK, COPY <MYDISK>MY.DIR>MY.FILE>6 <REDISK>YOUR.DIR>YOUR.FILE>2
OK,
```

Copying an Open File

If the source object is a file that is open, the file cannot be copied and the following error message is returned:

```
OK, copy tom>cat manx
File in use. Unable to open for reading "TOM>CAT" (copy)
ER!
```

Specifying a Nonexistent Object

If **source-pathname** does not exist, the COPY procedure aborts and returns an error message, as in the following example:

```
OK, copy erewhon>no_exist
Not found. "EREWHON>NO_EXIST" (copy)
ER!
```

Specifying a Target Pathname

target-pathname is the pathname that is given to the copied object (target object). Use an entryname to place the target object in your current directory:

```
COPY MONEY>DOLLAR PESO
```

In this example, the source object DOLLAR in the directory MONEY is copied to your current directory and given the name PESO.

Specifying an Existing Target Pathname

If the target directory already contains an object with the same name as **target-pathname**, COPY asks you if it should overwrite the existing target object with **source-pathname**. If you answer Y or YES (in either uppercase or lowercase), the existing target object is deleted and the copied object replaces it. The name of the copied object will be the same as the deleted object.

In the following example, COPY overwrites the file MONTH.DATA in the current directory with the file JUNE in the subdirectory SUMMER:

```
OK, COPY *>SUMMER>JUNE  MONTH.DATA
"MONTH.DATA" already exists, do you wish to overwrite it? YES
OK,
```

If the source object is a directory or access category, COPY first asks for your verification to continue the copy procedure. If you answer YES, COPY then asks you if you wish to overwrite the existing target object, as in the above example.

When overwriting an existing target object, you can use the NO_QUERY option to suppress all queries and replace the object. (The exception is if the target object was delete-protected with the SET_DELETE –PROTECT command. In this case, you also have to specify the –FORCE option.)

Caution

A common error when copying a source file into a directory is to specify only the directory's name as the target pathname, rather than as a part of the copied object's pathname. This could result in the deletion of the directory and its replacement by the copied object. For example, the correct command to copy your file GOLD into the directory METAL is

```
COPY GOLD METAL>GOLD
```

If you use the incorrect command COPY GOLD METAL, the following query prompt appears:

```
"*>METAL" (dir) already exists, do you wish to overwrite it?
```

If you answer YES or Y, COPY overwrites the directory METAL with the file GOLD, which it subsequently renames METAL.

To correctly copy the file GOLD to the directory METAL, you can also use the double-equal (==) name generation symbol, as follows:

```
COPY GOLD METAL>==
```

See Chapter 4 of this guide for more information on name generation.

Nested Directory Copying

COPY prevents nested directory copying, that is, copying a portion of a directory to the same portion of the directory. The following example shows the error message returned in this situation:

```
OK, COPY MYUFD>MY.DIR MYUFD>MY.DIR>MY.SUB.DIR
Source and target have duplicate/nested paths. (copy)
OK,
```

Use the –NO_CHECK option to suppress this check for nested directory copying.

Copying to an Open Target File

The COPY command cannot work if **target-pathname** is an open file. The following example shows the error message returned in this situation:

```
OK, copy webb webb2
"WEBB2" already exists, do you wish to overwrite it? yes
File open on delete. Unable to delete file "WEBB2" (copy)
ER!
```

The exception to this rule is if you are using the COPY command to replace an open EPF file with another EPF file. For this procedure, see the section below, Replacing a Mapped EPF File.

Omitting the Target Pathname

If you do not specify a target pathname, the source object is copied into your current directory with the entryname (the final element) of **source-pathname** as its name. The following example shows this format:

```
OK, COPY COLORS>BLUE
OK,
```

In this example, the source object BLUE in the directory COLORS is copied into your current directory and given the name BLUE.

Using the –REPLACE Option

The –REPLACE option replaces an existing target object with a source object without asking for your verification to overwrite it. The operation of this option, therefore, is similar to specifying the –NO_QUERY option in the overwrite procedure described in the section above, Specifying an Existing Target Pathname.

For the –REPLACE option to work, an object must exist with the same name as **target-pathname**. If **target-pathname** does not exist before you issue the COPY command, the command does nothing.

The –REPLACE option first deletes the existing object named **target-pathname** and then copies the source object under the name of the deleted object. Thus, you have replaced an old object with a new object but kept the old name.

The following command illustrates this option:

```
OK, COPY WILSON>STATUS  UPDATE -REPLACE
```

In this example, your file UPDATE is deleted and then WILSON>STATUS is copied into your directory under the name UPDATE. The original WILSON>STATUS file is unaffected.

If the source object is a directory or access category, COPY first asks for your verification to copy the object. Use the –NO_QUERY option to suppress this question.

If the target object is delete-protected (by the SET_DELETE command), use the –FORCE option in addition to –REPLACE, as follows:

```
OK, COPY WILSON>STATUS  UPDATE -REPLACE -FORCE
```

If **target-pathname** is an open file, the copy procedure does not work and returns an error message:

```
OK, copy lion tiger -replace
File open on delete. Unable to delete file "TIGER" (copy)
ER!
```

Using the –MERGE Option

The –MERGE option merges the source object with the target object. The –MERGE option may be used with directory and non-directory file system objects. For example, suppose you wish to perform an install update of a multilevel directory tree using a similar but separate directory tree. The –MERGE option allows you to do this without first having to delete the target tree and then copy and recreate the entire target tree from the source tree.

If both objects being merged are directories, the source tree is merged into the target tree. If you merge objects and only the source object is a directory, the target file is overwritten and truncated after the copy operation has been completed. If the target directory does not exist, the source object is simply copied.

If both objects being merged are non-directory objects, the source object is copied to the target object and the target object is truncated after the last record is copied, as in the following example:

```
OK, SLIST FILE.1
This is File #1
OK, SLIST FILE.2
This is File #2
OK, COPY FILE.1 FILE.2 -MERGE
OK, SLIST FILE.2
This is File #1
OK,
```

If the target file does not exist, the source file is simply copied.

If the target object type is different from the source object type (SAM, DAM, or CAM) and the target object is being changed during the copy, no merge takes place. The target object is first deleted, and the source object is then copied. To use the –MERGE option, you must have at least ALURW rights to the target object, since the target object (in most cases) is not deleted initially.

Note

Merging segment directories is similar to merging directories and file objects. Be aware that merging a source segment directory containing objects 1,2,3 with a target segment directory containing objects 2,3,4 results in a segment directory containing objects 1,2,3,4; the original target objects 2 and 3 are overwritten. When you perform a merge operation in which the source and target segment directories have the same name, the target segment directories are overwritten; you are not prompted to verify the operation.

Using the –SAVE_UFD Option

The –SAVE_UFD option is used in conjunction with the –INC option to perform incremental copy operations. This option instructs COPY always to copy directories, whether or not they have been modified.

Modifying a file directly affects the dump bit switch on the file's parent directory as well as the dump bit switch on the file itself. For instance, modifying the file BOOKS>TECH>FIRST causes the dump bit switch to be reset on both BOOKS>TECH and BOOKS>TECH>FIRST. The following command line requests an incremental copy operation on the file's grandparent (top-level) directory:

```
COPY BOOKS <BACKUP>BOOKS -INC
```

No copy takes place unless the dump bit switch on BOOKS is reset.

The –SAVE_UFD option disables the incremental check for directories only. This option causes the directory structure (its "skeleton") to be copied, even if no files within that directory are incrementally copied. In effect, the –SAVE_UFD option forces a treewalk of all directories; files that should be incrementally copied cannot be missed during an incremental copy. The example below illustrates the SAVE_UFD option:

```
OK, COPY <TDISK>T.UFD>@@ *>BACKUP.DIR>== -INC -SUFD
Ok to copy directory "<TDISK>T.UFD>T.DIR" to "*>BACKUP.DIR>T.DIR"? Y
"<TDISK>T.UFD>T.DIR" copied to "*>BACKUP.DIR>T.DIR".
OK,
```

COPY Options

The COPY options are listed below. You may specify two or more options if they do not conflict.

Option	*Meaning*
–ADD	Copies the source object to the target directory under the name **target-pathname**. This option works only if an object named **target-pathname** does not already exist in the target directory. –ADD is incompatible with –MERGE, –REPLACE, and –INCREMENTAL.
–CAM	Converts all copied DAM or SAM files to CAM files. **CAM** (Contiguous Access Method) files are files that contain groups of contiguous blocks. The default is preservation of the original file type. The –CAM option is used to copy one directory level, it does not copy subentries.
–COPY_ALL	Copies all the attributes of the source object, except its Access Category (ACAT), to **target-pathname**. (To copy an ACAT, specify that ACAT by itself on the COPY command line.) If possible, a specific ACL is placed on **target-pathname** so that its protection is the same as the source object. If you do not specify this option or another attribute-copying option (–DTM, –PROTECT, –QUOTA, or –RWLOCK), the target objects attributes are set to the protection defaults of the target;s directory, and no access categories in a subdirectory are copied to the target directory. To use this option on ACL directories, you must have Protect (P) access.
–DAM	Converts all copied SAM or CAM files to DAM files. The default is preservation of the original file type. The –DAM option is used to copy one directory level, not subentries.
–DELETE	Deletes the source object after it has been copied. (Default is no deletion.) If the source object is delete-protected, this option does not work unless you also specify the –FORCE option. You must have Delete (D) access on the source directory to use this option.

–DTM	Copies the date/time modified stamp of the source object to the target object (including subentries in a copied directory). (Default is to reset the date/time modified stamp to the current date/time.) To use this option with ACL directories, you must have Protect (P) access.
–FORCE	Deletes a source object (when used with the –DELETE option) or a target object (when used with the –REPLACE option) that is delete-protected (by the SET_DELETE command). If you do not specify –FORCE, COPY asks for your confirmation before deleting a delete-protected object, unless you specified the –NO_QUERY option (in which case the protected object is not deleted). The –FORCE option is particularly useful for overwriting a directory tree that contains delete-protected objects.
–HELP	Displays a list of COPY options and their usage.
–INCREMENTAL	Copies only source objects whose dump bit is off: that is, objects that have not been dumped to tape. (Default is to copy objects regardless of their dump bit settings.) This option works on or in directory objects and is propagated throughout a tree copy. (The –INC option checks the dump bit of the directory before checking individual files during a tree copy. If that directory's dump bit is on, no file objects in that directory are examined. Therefore, you should use the –SUFD option in conjunction with the –INC option when performing incremental backups.)
–LEVELS [n]	Copies down to the level indicated by **n** when you are copying a directory tree. **n** is a decimal integer from 0 to 999. If you omit –LEVEL, the entire tree is copied. If you omit **n**, 0 levels (the default value) are copied. That is, if you omit **n**, only the top-level directory is copied; none of the directory's subdirectories are copied.
–MERGE	Merges the source object with the target object. The –MERGE option may be used with directory and non-directory file system objects.
–MXL **n**	Allows you to set the extent length of a CAM file on a per-file basis. **n** is in the range 0 through 32768. If you set this value so that it is not 0, the default algorithm for allocating CAM files is no longer used for that file.
–NO_CHECK	Specifies that COPY should not check that the source and target objects are the same object. In addition, –NO_CHECK omits the validation that the directory object pathnames are not nested. This option is intended to be used only when performance is a major consideration; if you use the –NO_CHECK option, you must check to ensure that the source and target objects are not nested and are not the same object.
–NO_CMLV	Prevents the COPY command from invoking a new command level when the conditions "disk full" or "maximum quota exceeded" are encountered during the copy operation. If either condition occurs, the copy operation ends and an error message is displayed. The –NO_CMLV option is useful primarily for allowing a CPL program to handle a return error properly.
–NO_QUERY	Does not ask you to verify whether a copy procedure should be carried out during an unexpected or potentially dangerous situation. (Two examples of such requests are whether to overwrite an existing

object with the same name as **target-pathname** and whether to copy a directory or access directory.) If you specify this option, COPY carries out the copy procedure if it can. (Default is –QUERY.)

–PROTECT Copies the protection attributes (protection keys, passwords, and ACLs) of the source object to the target object. (Default is to use the default ACLs of the target directory and not to copy access categories in a source subdirectory.) To use this option on ACL directories, you must have Protect (P) access.

–QUERY Specifies that you must be asked to resolve any unexpected or potentially dangerous situations during a copy procedure. (Default)

–QUOTA Sets the maximum quota of a copied directory and its subdirectories the same as the source object. (Default is to place no quota on the target directory.) To use this option on ACL directories, you must have Protect (P) access.

–REPLACE Deletes **target-pathname** and then copies the source object to the target directory under the name **target-pathname**. The copied object replaces the existing target object, but retains the name of the replaced object. This option works only if an object named **target-pathname** already exists in the target directory.

–REPORT Reports the results of each successful copy operation.

–RWLOCK Sets the concurrency lock setting of the target object the same as that of the source object. (Default is to set the read/write locks to the system default.) To use this option on ACL directories, you must have Protect (P) access.

–SAM Converts all copied DAM or CAM files to SAM files. The default is preservation of the original file type. The –SAM option is used to copy one directory level, not subentries.

–SAVE_UFD Performs incremental copy operations in conjunction with the –INC option. This option instructs COPY always to copy directories, whether or not they have been modified.

Using Command-line Features

Command-line processing features such as iteration, wildcards, and name generation are particularly useful with COPY. The example below uses three command-line features:

```
OK, COPY (@.LIST @.BIN) ARCHIV>(=.OLDLIST =.OLDBIN)
```

This example does the following: copies all files with suffixes .LIST or .BIN in the current directory to the directory ARCHIV; replaces the suffix .LIST with .OLDLIST and the suffix .BIN with .OLDBIN; and preserves the rest of each entryname. For instance, a file called PAYROLL.BIN in the current directory is copied to the directory ARCHIV and is renamed PAYROLL.OLDBIN.

See Chapter 4 of this guide for further information on command-line features.

Replacing a Mapped EPF

As described above in the section Copying to an Open Target File, the COPY command normally does not work if the target object is an open file. The exception is if the target file is an open Executable Program Format (EPF) file. (An open EPF file is also called a mapped EPF. See the LIST_EPF command for definitions of mapped and unmapped EPFs.)

The file to replace the mapped EPF must be another EPF file. The second EPF file can be mapped or unmapped.

After you have replaced the EPF, any user who invokes it gets the new version. The old version, however, remains mapped in the address space of anyone who was using it when you replaced it. You should therefore inform all users of your change and suggest that they use the REMOVE_EPF command and then invoke the new version.

Note

Do not confuse the replace-EPF operation with the –REPLACE option described above. Unlike the –REPLACE option, the replace-EPF operation does not delete the replaced EPF but instead renames it.

Specifying Source and Target Pathnames

The COPY command format for the replace-EPF operation is

COPY **source-pathname target-pathname [options]**

source-pathname is the EPF that replaces the mapped EPF file. **target-pathname** is the mapped EPF file that you want replaced. **target-pathname** must have a .RUN suffix.

The following example illustrates a command line for a replace-EPF operation:

```
OK, COPY MYLIB>NEW_EPF.RUN LIBRARIES*>OLD_EPF.RUN
```

OLD_EPF (**target-pathname**) is the mapped EPF file you want to replace and NEW_EPF (**source-pathname**) is the file you are putting in its place.

How the Replace-EPF Operation Works

The replace-EPF operation works as follows:

1. The name of the mapped EPF file (**target-pathname**) is changed by having its .RUN suffix replaced by a .RP**n** suffix (where **n** is a digit from 0 to 9, inclusive). The file is not deleted, but becomes a REPLACE file. The digit to be assigned to .RP**n** depends on how many REPLACE files exist. (Only 10 REPLACE files can exist at any one time.) In the above example, OLD_EPF.RUN might be renamed OLD_EPF.RP2.
2. The source EPF file (**source-pathname**) is then copied to **target-pathname**. In the example, NEW_EPF.RUN is copied to LIBRARIES*>OLD_EPF.RUN.

By default during the replace-EPF operation, COPY asks you if it should replace **target-pathname**. If you answer YES, COPY informs you when the operation is completed and then displays the new name of the replaced file. These messages are shown in the following example:

```
OK, COPY TIM>FORTRAN_IO_LIBRARY.RUN LIBRARIES*>==
Ok to replace EPF file LIBRARIES*>FORTRAN_IO_LIBRARY.RUN? YES
New version of EPF file FORTRAN_IO_LIBRARY.RUN now in place.
Old version of active EPF file now named FORTRAN_IO_LIBRARY.RP0.
```

Use the –NO_QUERY option to suppress the verification query and replace the EPF automatically.

If 10 REPLACE files already exist, COPY first asks you whether it can delete one of the REPLACE files that is not currently mapped. If you answer YES, the operation continues, as shown below:

```
OK, COPY TIM>LD.RUN CMDNC0>LD.RUN
Ok to replace EPF file CMDNC0>LD.RUN? YES
Ok to delete EPF file CMDNC0>LD.RP2? YES
New version of EPF file LD.RUN now in place.
Old version of active EPF file now named LD.RP2.
```

Use the –NO_QUERY option to suppress both prompts.

If all 10 REPLACE files are mapped when you issue the command, the replace-EPF operation cannot be completed, as the following example illustrates:

```
OK, COPY TIM>LD.RUN CMDNC0>LD.RUN
Ok to replace EPF file CMDNC0>LD.RUN? YES
EPF replace files are all in use.
Unable to complete file copy. (copy)
```

For more information on EPFs and on using the COPY command to replace mapped EPFs, see the *Programmer's Guide to BIND and EPFs*.

Copying With Password Directories

Access requirements are different under password directories. In all cases, you must have owner access on the target object. To delete an object, Delete (D) access is required for the object.

The password protecting a source object is also copied if you specify –PROTECT. Protection attributes include protection keys (for files, directories, and segment directories) and passwords (for directories only). If you do not specify –PROTECT, the source object is copied with the system default rights of RWD NIL, which means that the owner has all rights and nonowners have none.

To copy the passwords of a directory, you must have owner rights in the source object. If you do not have owner rights, the copied directory acquires the system default passwords. For owner passwords, the default is blank. For nonowners, the default provides no password (that is, the password is null).

Note

COPY does not allow the MFD, BOOT, or DSKRAT files of an MFD to be overwritten or copied. To copy a boot file to an MFD, you must first restore the new boot to memory and then save it under the name BOOT. This restriction does not apply when these files exist somewhere other than in an MFD.

▶ COPY_DISK [options]

Operator Command

COPY_DISK copies physical disk partitions to other physical partitions. For details on the operation and options of COPY_DISK, see the *Data Backup and Recovery Guide*.

▶ COPY_RBF source-pathname target-pathname [options]

COPY_RBF copies a ROAM file. Use COPY_RBF to move or replicate a ROAM file. When you move a ROAM file, the original source file is deleted after a successful copy. When you replicate a file, both source and duplicate files remain.

source-pathame is the name of the existing master or slave segment. If you are moving the file, this name can be a subfile name. If you are replicating the file, **source-pathname** must be the master segment directory. **target-pathname** is the pathname of the new location for the source file.

COPY_RBF has the following options:

Option	*Meaning*
–CAM	Creates all the subfiles of the target segment directory as contiguous files. Cannot be used with the –DAM option. If neither –CAM nor –DAM is specified, the subfiles of the target segment directory assume the same file type as those of the source segment directory.
–DAM	Creates all the subfiles of the target segment directory as DAM files. Cannot be used with the –CAM option. If neither –DAM nor –CAM is specified, the subfiles of the target segment directory assume the same file type as those of the source segment directory.
–DELETE	Deletes the source file after a successful copy. The target file retains the ROAM file identifier of the original source file. If –DELETE is not specified, the new file is assigned a new ROAM file identifier. –DELETE cannot be used if **target-pathname** is on a remote system.
–PROTECT	Assigns to the target file the same ACL protection that the source file possesses.

–REPORT Reports each RBF subfile as it is copied or deleted. The default is not to report successful operations on individual subfiles. With or without the –REPORT option, COPY_RBF reports the successful copy of the entire RBF.

For details on COPY_RBF and on ROAM files, see the *ROAM Administrator's Guide*.

▶ CPL pathname

CPL invokes the Command Processor Language (CPL) interpreter and executes a CPL program.

pathname is the name of the file containing the CPL program. CPL searches for and executes the file in the following order:

1. If **pathname** has a .CPL suffix, the file is executed.
2. If **pathname** does not have a .CPL suffix, CPL searches for a file named **pathname**.CPL and runs it if it exists.
3. If **pathname**.CPL does not exist, CPL then searches for **pathname** without the .CPL suffix and runs it when found.

For more information on CPL, see the *CPL User's Guide*.

See also RESUME.

▶ CPMPC pathname [–CRn] [–PRINT]

CPMPC punches the contents of a file onto cards in an assigned card reader/punch.

You may specify the CPMPC parameters in any order. **pathname** is the name of the file to be punched. The value of **n** in the –CR option is 0 or 1, depending on whether you specify the first (CR0) or second (CR1) card reader/punch. You must assign the card reader/punch with an ASSIGN CR0 or ASSIGN CR1 command.

The –PRINT option causes punched data to be printed on the card if the punch has that capability.

CPMPC does not punch an end-of-file ($E) card at the end of the output deck of punched cards. For further information on $E cards, see the CRMPC command.

▶ CPW

See CHANGE_PASSWORD.

▶ CRASH_AUDIT –MT n –DUMPFILE [pathname] –OUTFILE [pathname] –MAP [map_pathname]

Operator Command

CRASH_AUDIT invokes a utility that retrieves audit records of memory buffers as they existed at the time of a system halt. CRASH_AUDIT may be invoked by the System Administrator or may be invoked from the supervisor terminal. For more information, see the *System Administrator's Guide, Volume III: System Access and Security*.

▶ CREATE pathname [options]

CREATE creates a new directory. **pathname** is the name of the new directory. If **pathname** is an entryname, the directory is created in the current directory.

The CREATE options are listed below. You cannot use the –PASSWORD option with the –MAX and –CATEGORY options.

Option	*Meaning*
–CATEGORY categoryname	Protects the directory with the access category named **categoryname**. The access category must reside in the same directory as the newly created directory.
–MAX n	Creates the directory as a quota directory and sets the quota to **n**. **n** is a decimal integer between 0 and the maximum number of records on the disk.
–PASSWORD	Creates the directory as a password directory. If you do not specify –PASSWORD, the directory is the same type as its parent directory.

Rules for Creating Directories

- You must have Add (A) rights to create a new directory within an ACL directory.
- If the new directory is created subordinate to an ACL directory, the new directory becomes an ACL directory. It is protected by default protection taken from its parent directory.
- To create a password directory subordinate to an ACL directory, use the –PASSWORD option.
- If the new directory is created subordinate to a password directory, it is automatically created as a password directory, whether or not you specify the –PASSWORD option. The directory is created with a blank owner password and a null nonowner password. (Any password will match it.) The protection keys are set to RWD NIL, which allow Read, Write, and Delete rights to the owner and no rights to nonowners.
- An ACL directory cannot be subordinate to a password directory.

▶ CREATK

CREATK invokes an interactive program that creates, modifies, and monitors keyed-index or direct-access files.

CREATK is a part of the MIDASPLUS subsystem. When invoked, CREATK sets up a dialogue that asks you questions about the template structure. For further information, refer to the *MIDASPLUS User's Guide*.

▶ CRMPC pathname [–CRn] [–PRINT]

CRMPC reads cards from the parallel interface card reader connected to the MPC controller and loads card image ASCII data into the file specified by **pathname**.

You may specify the CRMPC parameters in any order. The –CR option specifies the number of the card reader/punch. **n** must be 0 or 1.

The –PRINT option prints the contents of each card on the card, if the card reader has that capability.

Reading of the cards continues until one of the following occurs:

- A card is read that has $E in columns 1 and 2 (the recommended way to stop).
- There are no more cards in the reader.
- The STOP button on the card reader is pressed.
- BREAK or CONTROL-P is pressed on the terminal.

Cards are expected to be in 029 (EBCDIC) representation. Control cards may be inserted into the card deck to instruct the card reader, as follows:

Columns 1 and 2 of deck control card	*Instruction*
$6	Placed before a deck of cards in 026 (BCD) format. Instructs the card reader to interpret 026 cards as if they were in 029 format.
$9	Instructs the card reader to resume reading in 029 format.
$E	Placed last in the deck and signals the end of the deck. Control returns to PRIMOS and the file is closed.

▶ DATE [option]

DATE displays the current calendar date and clock time.

If DATE is invoked with no option, it uses the following date format:

```
OK, DATE
14 July 87 11:05:40 Tuesday
OK,
```

To obtain a different format, use one of the following options. The sample outputs use July 14, 1987 as the date.

Option	*Sample Output*
-AMPM	11:05 AM
-CAL	July 14, 1987
-DAY	14
-DOW	Tuesday
-FTAG	840714.110508
-FULL	84-07-14.11:05:08.Tue
-MONTH	July
-TAG	870714
-TIME	11:05:08
-USA	07/14/87
-UFULL	07/14/87.11:05:08.Tue
-VFULL	14 July 87 11:05:08 Tuesday
-VIS	14 July 87
-YEAR	1987

DATE is particularly useful to date command output files.

▶ DBACP

DBACP invokes the Database Administrator Command Process, which allows the Database Administrator to create, investigate, and change the various file structures within databases. For details, refer to the *DBMS Administrator's Guide*.

▶ DBASIC [pathname]

DBASIC invokes the Prime version of interpretive BASIC that has double-precision arithmetic capabilities. The operation of the DBASIC command is the same as the operation of the BASIC command.

▶ DBG pathname [options]

DBG invokes the Prime Source Level Debugger.

The Source Level Debugger is an interactive debugger program for programs written in C, COBOL 74, FORTRAN IV, FORTRAN 77, Pascal, PL/I Subset G, and V-mode RPG II. **pathname** is the name of an executable runfile. The runfile's source program must be compiled with the –DEBUG option, and its object (binary) program must be loaded (by BIND or SEG) and executed.

When invoked, the Debugger displays a right angle-bracket (>) prompt and waits for a Debugger command. Enter HELP for online help.

For a full discussion of the operation and options of DBG, see the *Source Level Debugger User's Guide.*

See also HPSD; PSD; PSD20; VPSD; VPSD16.

▶ DBUTL

DBUTL invokes a database dump utility that allows users to monitor the contents of a database schema and shared user table. When invoked, DBUTL displays a colon (:) prompt and waits for a DBUTL command, which must be entered in uppercase. For details on DBUTL, consult the *DBMS Administrator's Guide.*

▶ DEFINE_GVAR { pathname [–CREATE] | –OFF }

DEFINE_GVAR creates and manipulates global variable files.

You must create a global variable file before you can define global variables. Yo may have more than one global file, but only one can be active at any time.

You can use the DEFINE_GVAR command at command level or inside a CP program.

Creating a Global Variable File

To create a global variable file, use the format

DEFINE_GVAR pathname –CREATE

pathname specifies the name of the file, which is empty at creation. The command also activates the file.

The following example creates and activates an empty global variable file named GLOBAL.VARS:

```
OK, DEFINE_GVAR  GLOBAL.VARS  -CREATE
```

Activating an Existing Global Variable File

You must activate the global variable file at the beginning of any work session during which you want to use the global variables it contains. To activate the file, use the format

DEFINE_GVAR pathname

pathname is the name of the file.

If you have a password on the directory containing your global variable file, you must use a full pathname with the password and you must enclose the entire pathname in single quotation marks. In the following example, SECRET is the password for the directory BEECH:

```
OK, DEFINE_GVAR '<FOREST>BEECH SECRET>GLOBAL.VARS'
```

When the file is active, you can add to, delete, list, and make use of any variables it contains.

Deactivating a Global Variable File

To deactivate an active global variable file, use the format

DEFINE_GVAR –OFF

Global variable files are also deactivated when you log out or when you activate a new global variable file with DEFINE_GVAR.

Deleting a Global Variable File

To delete a global variable file, deactivate it with the command DEFINE_GVAR –OFF and then use the DELETE command, as in the following example:

```
OK, DEFINE_GVAR GLOBAL.VARS -OFF
OK, DELETE GLOBAL.VARS
OK,
```

See also DELETE_VAR; LIST_VAR; SET_VAR.

▶ DELAY [min [max [margin]]]

DELAY defines a time function that delays the printing of a character after a carriage return (CR) has been output to a terminal.

The values specified for DELAY must be octal numbers. The meanings of the DELAY parameters are

Parameter	*Meaning*
min	Defines the number of character-times (the time it takes the system to type a character on a line) to delay when CR is output at the left margin. (Default is 6.)
max	Defines the number of character-times to delay when CR is output at the right margin. (Default is 14 octal, 12 decimal.)
margin	Defines the number of characters required to move to the right margin. (Default is 110 octal, 72 decimal.)

If a carriage return is output at some point within a line, the time delay is proportional to the number of characters typed. The defaults given are for a 30 characters-per-second (cps) terminal. These defaults are assumed if you issue DELAY with no parameters.

The following example illustrates the DELAY command:

```
OK, DELAY 0 10 100
```

To turn off delay padding, use the following command:

```
OK, DELAY 0 0 1
```

You can issue the DELAY command before you log in. DELAY can also be issued from the supervisor terminal if the terminal is designated to be User 1 (by using either the USRASR operator command or the VCP command MO USER).

DELAY is particularly useful for a terminal with a nonstandard line speed. In this case, the command DELAY 10 should set the terminal to function in the Prime computer configuration.

The DELAY command works with terminals on Network Terminal Service (NTS) lines as well as with those on local lines. You can use DELAY before you log in to a terminal with an NTS line, provided that the NTS line was connected previously.

▶ DELETE pathname [options]

DELETE deletes file system objects.

pathname (the target object) is the name of the file, directory, segment directory, or access category you want to delete. You can specify an entryname if the object is in your current directory.

To use DELETE on an ACL directory, you must have Delete (D) access on the target object's parent directory. To delete an object from a password directory, you must have Delete (D) access on the target object or owner access on the directory.

If **pathname** is a directory or an access category, PRIMOS first asks you to verify the object's deletion, as in the following example:

```
OK, DELETE CASE
Ok to delete directory "CASE"? YES
OK,
```

Use the –NO_QUERY option to delete the object without the verification query.

DELETE Options

You may specify one or more options in any order on the command line following the pathname. The options are

Option	*Meaning*
–FORCE	Deletes an object that is delete-protected by the SET_DELETE command. Use –FORCE to delete a directory that may contain delete-protected objects. If you do not specify –FORCE, PRIMOS asks you to confirm the deletion of a delete-protected object.
–NO_QUERY	Instructs PRIMOS to resolve any unexpected situations during a deletion procedure. If you specify this option for a delete-protected object but do not specify –FORCE, the object is not deleted.
–QUERY	Instructs PRIMOS to ask you to resolve any unexpected situations during a deletion procedure. (Default)
–REPORT	Reports the results of each successful deletion.

Using Command-line Features With DELETE

Some command-line features, such as wildcarding and iteration, are particularly useful with the DELETE command.

Using Wildcards With DELETE

When your pathname includes wildcards, PRIMOS asks you to verify the deletion of each file system name that matches the wildcard name, even if you specified the –NO_QUERY option. The reason for this is that –NO_QUERY is a DELETE option and is overridden by the verification procedure of wildcards.

To suppress wildcard verification queries, use the wildcard –NO_VERIFY option (abbreviated –NVFY) with the command, as in the following example:

```
OK, DELETE TE@@ -NO_VERIFY
```

With –NO_VERIFY, PRIMOS queries you about the deletion of subdirectories and access categories, but not about files and segment directories.

To suppress all queries about deletions, specify both the –NO_QUERY and –NO_VERIFY options, as in the following example:

```
OK, DELETE TE@@ -NO_VERIFY NO_QUERY
```

Using Iteration Lists With DELETE

To delete two or more specific files, substitute an iteration list for the pathname, as follows:

```
OK, DELETE (PEAR PLUM *>CARROT>ROOT)
```

This example deletes the objects PEAR and PLUM from your current directory and the object ROOT in the subdirectory CARROT.

For more information on command line features, see the *PRIMOS User's Guide* and Chapter 4 of this guide.

Examples of Using DELETE

The following examples illustrate the DELETE command.

Example 1: Deleting a delete-protected file without options:

```
OK, DELETE JONES.REPORT
"JONES.REPORT" protected, ok to force delete? YES
OK,
```

Example 2: Deleting a delete-protected file with the –NO_QUERY option:

```
OK, DELETE BROWN.MEMO -NO_QUERY
File is delete-protected.  Unable to delete "BROWN.MEMO" (delete)
ER!
```

Example 3: Deleting a delete-protected file with the –FORCE and –REPORT options:

```
OK, DELETE SMITH.RPT -FORCE -REPORT
"SMITH.RPT" deleted.
OK,
```

Example 4: Deleting files with a wildcard name but without the –NO_VERIFY option:

```
OK, DELETE RE@@
(std$cp) Verify wildcard selections for "RE@@":
"REBATE"? YES
"REPORT.MAY"? YES
"RESULTS" (dir)? YES
Ok to delete directory "RESULTS"? YES
OK,
```

Example 5: Same as Example 4, but using the –NO_VERIFY option:

```
OK, DELETE RE@@ -NO_VERIFY
Ok to delete directory "RESULTS"? YES
```

Example 6: Same as Example 5, but also using the –NO_QUERY option:

```
OK, DELETE RE@@ -NO_QUERY -NO_VERIFY
OK,
```

Note

DELETE cannot delete the MFD, BOOT, or DSKRAT files in an MFD. These files can be deleted only if they exist somewhere other than in an MFD.

▶ DELETE_RBF pathname [options]

DELETE_RBF deletes an active or inactive ROAM file (master and slaves).

pathname is the name of the file to be deleted. It should be the name of the master segment directory.

The DELETE_RBF options are

Option	*Meaning*
–NO_QUERY	Deletes the file without asking for your verification.
–REPORT	Reports each RBF subfile as it is deleted. The default is not to report successful deletions of individual subfiles. With or without the –REPORT option, DELETE_RBF reports the successful deletion of the entire RBF.

For details on DELETE_RBF and on ROAM files, see the *ROAM Administrator's Guide*.

▶ DELETE_VAR variable-name-1 [...variable-name-n]

DELETE_VAR deletes one or more global variables from an active global variable file.

Before you can use DELETE_VAR, you must activate the global variable file with the DEFINE_GVAR command. **variable-name** can be a wildcard name. If you specify more than one global variable, you must separate the names with a space or a comma.

Example 1: This example activates the global variable file GLOBAL.VARS and deletes the variable .ALPHA:

```
OK, DEFINE_GVAR GLOBAL.VARS
OK, DELETE_VAR .ALPHA
OK,
```

Example 2: This example deletes three variables from the active global variable file:

```
OK, delete_var .leo .virgo .pisces
OK,
```

See also LIST_VAR; SET_VAR.

▶ DELSEG { DELSEG seg-number-1 [–TO seg-number-2] / ALL }

DELSEG frees (deletes) segments.

seg-number is the segment number of the segment to be freed. It must be specified in octal and must be 2000_8 or above for User 1 (4001_8 and above for all other users). The DELSEG command can be used to delete segment 4000_8 but the results are unpredictable. Therefore, using DELSEG to delete segment 4000_8 is not recommended.

The –TO option deletes all the segments in a range of numbers, as in the following example:

```
OK, DELSEG 4003 -TO 4050
```

The DELSEG ALL format deletes all segments belonging to the user at that terminal.

A **BAD PARAMETER** message is displayed if you specify an illegal segment number. Deleting a nonexistent segment has no effect.

▶ DEVICE_ACLS {-ON / -OFF}

Operator Command

DEVICE_ACLS sets Access Control List (ACL) rights on devices such as tape drives and printers. This command may be issued only at the supervisor terminal or by a user logged in as the System Administrator. For more information, see the *System Administrator's Guide, Volume III: System Access and Security.*

▶ DIAG

DIAG invokes the PRISAM File Diagnostic Utility. The File Diagnostic Utility (DIAG) is an interactive program that verifies the structural validity of both relative and indexed PRISAM files. When invoked, DIAG displays a **DIAG>** prompt and waits for you to enter a DIAG subcommand. Type HELP for a list of the DIAG subcommands. For details on DIAG, see the *PRISAM User's Guide.*

▶ DISCOVER [-CLUP]

DISCOVER invokes the DBMS and PRISAM query language and report writer.

Use this query language, in conjunction with DBMS and PRISAM, to retrieve information from DBMS databases and PRISAM indexed files and to update PRISAM indexed files without writing application programs. You can retrieve information by typing nonprocedural statements, and you can also format the retrieved information by using the DISCOVER Report Generator if the default display format is not what you want.

The -CLUP option activates a cleanup utility that allows you to recover manually from a software error; -CLUP cleans up any internal inconsistencies and subsequently returns control to command level.

After you issue the DISCOVER command, the program displays a right angle-bracket (>) prompt and waits for a DISCOVER command, as follows:

```
OK, DISCOVER
[DISCOVER/DBMS&PRISAM Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]

>
```

Type HELP to display information on DISCOVER commands and on the Report Generator.

For detailed information on DISCOVER, see the *DISCOVER Reference Guide* and the *DISCOVER User's Guide.*

▶ DISKS [NOT] pdev-0 [...pdev-7]

Operator Command

DISKS adds or removes the specified physical disk(s) to or from the Assignable Disks Table. For detailed information, see the *Operator's Guide to System Commands*.

▶ DISPLAY_LOG logname [logtype] [options]

Operator Command

DISPLAY_LOG either displays messages from Distributed System Management (DSM) logs at the desired user terminal or writes them to a separate disk file. DISPLAY_LOG is also used to read and print all Prime event logs, and replaces PRINT_SYSLOG and PRINT_NETLOG at Rev. 21.0. Access to DISPLAY_LOG is controlled by the System Administrator.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ DISTRIBUTE_DSM [options]

Operator Command

DISTRIBUTE_DSM copies a master Distributed System Management (DSM) configuration file to all nodes in a configuration group or to a named node or node group. DISTRIBUTE_DSM also provides a facility for removing nodes from a configuration. DISTRIBUTE_DSM command-line options allow you to set the terminal environment so that DISTRIBUTE_DSM works properly.

▶ DMSTK

See DUMP_STACK.

▶ DPTCFG input-file [options]

Operator Command

DPTCFG compiles the configuration file needed to set up a DPTX system. For details, see the *Distributed Processing Terminal Executive Guide*.

▶ DPTX { –DATA pathname | –ON | –OFF }

Operator Command

DPTX is used by the System Administrator to configure and enable the DPTX (Distributed Processing Terminal Executive) system, which allows the use of IBM 3271/3277 terminals as Prime terminals and/or the use of 3271/3277 terminals or OWL 1200 terminals attached to Prime as IBM host terminals. For full details, see the *Distributed Processing Terminal Executive Guide*.

▶ DPTXMTR [{ –QUEUE [–FREQUENCY seconds] | –TOTALS [–FREQUENCY minutes] }]

DPTXMTR displays information about the DPTX queues and events on the communication lines.

Use the –QUEUE option to monitor the DPTX queues. Use –TOTALS to display information about the communication line events. For detailed information, see the *Distributed Processing Terminal Executive Guide*.

▶ DROPDTR

DROPDTR drops the DTR (Data Terminal Ready) signal associated with an asynchronous line.

You cannot issue the DROPDTR command while logged in. The DROPDTR command is intended for use only in the following situation:

1. A user has been communicating with a Prime computer over a dialup asynchronous line, using a port selector or modem.
2. The user logs out.
3. The user now wants to disconnect from the current line and reconnect to a new line. (For example, the user may wish to log in to a different node on a network.) To force the disconnect, the logged-out user gives the DROPDTR command.

Normally, the DTR is dropped following a grace period of a maximum of 10 minutes. The length of the grace period is set by the System Administrator, using the AMLTIM or DTRDRP configuration directives.

▶ DUMP_SEGMENT [options]

Operator Command

DUMP_SEGMENT allows the operator to specify which additional segments for all users are written to tape during a tape dump. These segments, which are specified by their octal numbers, are in addition to the default segments. For more information, see the *Operator's Guide to System Commands*.

▶ DUMP_STACK [options]

DUMP_STACK produces a call/return trace of the user's command loop stack and static mode stack (if any).

The DUMP_STACK options, which may be given in any order, specify how the dump is to be done. Addresses are always printed in octal.

Option	*Meaning*
–ALL	Begins the dump with the frame from which DUMP_STACK was called. (Default is to begin the dump with the most recent condition frame.)
–BRIEF	Specifies a short format dump, omitting condition frames and fault frames. If you do not specify –BRIEF, the dump is printed in full format, which is the default.
–FRAMES n	Specifies that only **n** frames of the stack are to be dumped. **n** must be a positive decimal integer. The default is to dump the entire stack.
–FROM n	Begins the dump with frame **n**. (The frame from which DUMP_STACK is called is frame 1.) If you do not specify –FROM, the –ALL option determines the starting point for the dump.
–ON_UNITS	Produces a list of on-units established by each frame that is dumped.

For information on the format of the stack dump, see Appendix B.

▶ DUMP_USER { username-1 [... username-10] / –HELP }

Operator Command

DUMP_USER specifies which user segments are to be written to tape during a tape dump. The segment numbers must be in the range 4000 through 7777 octal. For more information, see the *Operator's Guide to System Commands*.

▶ E

See ELIGTS.

▶ ECL

See EDIT_CMD_LINE.

▶ ED [pathname]

ED invokes EDITOR, the Prime line-oriented text editor.

If you specify **pathname**, that file is loaded into EDITOR's text buffer and EDITOR is started in Edit mode. Use a filename if the file is in your current directory. EDITOR automatically opens and closes files and file units.

If you omit **pathname**, EDITOR is started in Input mode with an empty text buffer.

If you accidentally return control to PRIMOS (for example, by pressing the BREAK or CONTROL-P keys), you can restart EDITOR without losing any of the text buffer by issuing either of the following commands: START 1000 (continue from the break) or START 1001 (resume in Edit mode).

For details of EDITOR's operation, see the *New User's Guide to EDITOR and RUNOFF*.

▶ EDAC

See EDIT_ACCESS.

▶ EDB input-filename [output-filename]

EDB loads and starts EDB, the Prime binary editor.

EDB is used primarily for creating and maintaining libraries of subroutines. When EDB starts, it displays an **ENTER** prompt and waits for an EDB subcommand.

The input and output files may be on disk or paper tape. **input-filename** should be an existing library or the binary output of a Prime language translator. If **output-filename** is an existing file, it is overwritten by the new output file.

If paper tape is used for either the input or output file, use –PTR as the filename. In the following example EDB is started with a paper tape as the input file and NEWLIB as the name of the output file.

```
OK, EDB -PTR NEWLIB
```

If a terminal is used as either the input or output file, use –ASR as the filename. In the following example PASLIB2 is a PRIMOS input file and the output file is your terminal.

```
OK, EDB PASLIB2 -ASR
```

A PRIMOS input or output file is assumed if neither –ASR or –PTR is specified.

For details on EDB, see the *Advanced Programmer's Guide, Volume I: BIND and EPFs*.

▶ EDIT_ACCESS target-pathname acl [–NO_QUERY]

EDIT_ACCESS modifies an existing Access Control List (ACL).

target-pathname is the name of a file, directory, or segment directory protected by a specific ACL, or the name of an access category. To use EDIT_ACCESS, you must have Protect (P) access on **target-pathname**.

The ACL of **target-pathname** should have been created with the SET_ACCESS command. (Use the LIST_ACCESS command to display the contents of a specific ACL or access category.)

If **target-pathname** is an access category, you do not have to specify the .ACAT suffix unless there is another object in the directory with the same name as the unsuffixed access category.

acl specifies the access rights for an individual or a group. The format for **acl** is a list of names and access rights in the following format:

identifier-1:rights [... identifier-n:rights]

identifier is a user ID, a group name (which must begin with a period), or the special identifier $REST. **rights** are one or more of the mnemonic codes listed under the SET_ACCESS command. **identifier** and **rights** are separated by a colon but not by any blank spaces. For a complete description of identifiers and rights, see the SET_ACCESS command.

The –NO_QUERY option suppresses any queries and instructs PRIMOS to resolve any problems during the editing procedure.

Adding or Changing an Identifier

To add an identifier to a specific ACL or access category, or to change the access rights of an existing identifier, specify the identifier and the access rights as explained in the SET_ACCESS command. For example, the following command adds or changes the rights of users LYNN and ALFRED in the specific ACL of file DEPT_LIST:

```
OK, EDAC DEPT_LIST LYNN:ALL ALFRED:DALURW
```

When you are adding to or modifying an ACL, the other identifiers and their rights in the ACL are not affected.

Deleting an Identifier

To delete an identifier from a specific ACL or access category, specify the identifier with a null **rights** (that is, do not type anything after the colon). For example, the following command deletes user TOM from the access category BELL.ACAT:

```
OK, EDIT_ACCESS BELL.ACAT TOM:
```

When you are deleting with EDIT_ACCESS, the other contents of the ACL are not affected.

Using EDIT_ACCESS on Your Current Directory

To modify the ACL of your current directory (that is, the directory to which you are currently attached), use the full pathname of the directory. For example, if the name of your current directory is SAIL and its parent directory is MAST, use its pathname as follows:

```
OK, EDIT_ACCESS MAST>SAIL JOHN:ALL LARRY:LUR
```

If your current directory is a top-level directory, use its diskname. For example, if the top-level directory MAST is on the disk partition named SHIP, use the following format:

```
OK, EDIT_ACCESS <SHIP>MAST JOHN:ALL LARRY:LUR
```

Results of Using EDIT_ACCESS

Depending on the status of **target-pathname**, EDIT_ACCESS works as follows:

- If **target-pathname** is an existing access category, its contents are modified as specified by **acl**.
- If **target-pathname** is a file or directory protected by a specific ACL, the ACL is modified as specified by **acl**.
- If the file, segment directory, or directory is protected by a default ACL (that is, if it is not protected by a specific ACL or access category), EDIT_ACCESS assumes that you want to create a specific ACL for the object and asks for your verification:

```
OK, EDAC PLAN.B GREEN:ALL $REST:NONE
"PLAN.B" is default-protected.  Create specific ACL? YES
OK,
```

Answering Y, YES, or OK creates a specific ACL containing the specified **acl**, while any other answer aborts the command. If you specify the –NO_QUERY option, EDIT_ACCESS creates the specific ACL without asking for your verification.

- If **target-pathname** does not exist, EDIT_ACCESS assumes that you want to create an access category and asks for your verification:

```
OK, EDIT_ACCESS DEPT JAMES:ALL ANN:DALURW .TYPING:ALL
"DEPT.ACAT" does not exist.  Create access category? YES
OK,
```

Answering Y, YES, or OK creates an access category named **target-pathname**.ACAT while any other answer aborts the command. If you specify the –NO_QUERY option, EDIT_ACCESS creates the access category without asking for your verification.

Note

If the access rights on a directory are changed while you are attached to that directory, you must reattach to that directory before the new access rights affect you.

For additional information on ACLs, see the SET_ACCESS command or the *PRIMOS User's Guide*.

▶ EDIT_CMD_LINE [options]

EDIT_CMD_LINE (ECL) is a command-line editor that allows you to actively manage most aspects of terminal input.

ECL allows you to

- Edit terminal input after you have typed it in, but before you have actually submitted it for processing.
- Bind any key sequence to any command.
- Bind and program a maximum of 512 programmable function keys.
- Keep a command history of the last 200 commands issued. You may save and edit this command history.

ECL references ECL$LIB, a process-class library EPF that contains the actual command-line processing routines. When you invoke ECL, you actually replace the PRIMOS routine that reads and processes terminal input; thus, you should have some familiarity with the ECL facility before using it. For more information about ECL and its command-line characteristics, see Chapter 6. For more information about the use of ECL, refer to the *PRIMOS User's Guide*.

ECL Command-line Options

Table 2-1 lists all of the ECL options in separate categories by function. Refer to the table to find the category of the desired option, then find that option in the appropriate section in the pages that follow.

Some options act as toggle switches; that is, one option turns a condition off and an accompanying option turns the condition on. These options are listed together, separated by a slash (/) character, in the table. If a default value for an option is applicable, that value is listed in the right column.

TABLE 2-1
ECL Command Line Options

Category	*Option*	*Default Value*
Basic Options	–HELP	
	–INITIALIZE	
	–OFF / –ON	
	–RB / –EB / –WB;	(OK, ER! OK,)
	–VERSION	
Terminal Bindings & Characteristics	–BIND_TERM	(fundamental)
	–BPS / –SPEED	(9600 baud)
	–HEIGHT	(24 lines)
	–SAVE_TERM / –RESTORE_TERM	
	–TERMINAL_TYPE	(generic terminal)
	–WALLPAPER	
	–WIDTH	(80 columns)
	–XOFF / –NO_XOFF	–XOFF
Controlling Terminal I/O	–CLEAN_COMO / –NO_CLEAN_COMO	–CLEAN_COMO
	–CLEAR_LINE / –NO_CLEAR_LINE	–NO_CLEAR_LINE
	–EDIT_COMI / –NO_EDIT_COMI	–NO_EDIT_COMI
	–OBEY_ERKL / –NO_OBEY_ERKL	–OBEY_ERKL
	–ROW_MAJOR / –COL_MAJOR	–ROW_MAJOR
	–SILENT	
Command History	–SAVE_HISTORY / –RESTORE_HISTORY	
	–STACK / –NO_STACK	–NO_STACK
	–SHOW_HIDDEN / –NO_SHOW_HIDDEN	–NO_SHOW_HIDDEN
	–STICK / –NO_STICK	–NO_STICK
Pathname Completion	–WILD_ABBREV / –NO_WILD_ABBREV	–NO_WILD_ABBREV
	–WILD_DIRECTORY / –NO_WILD_DIRECTORY	–WILD_DIRECTORY
	–WILD_MENU / –NO_WILD_MENU	–WILD_MENU
	–WILD_TAIL / –NO_WILD_TAIL	–NO_WILD_TAIL
Editing Characteristics	–CASE_SEARCH / –NO_CASE_SEARCH	–NO_CASE_SEARCH
	–COMPONENT / –ENTRY	–COMPONENT

The command-line options of ECL are described in the sections below. The sections are arranged by function in the following categories:

- Basic options
- Terminal bindings and characteristics
- Controlling terminal I/O
- Command history
- Pathname completion
- Editing characteristics

Options that act as toggle switches (that is, one option turns a condition off and an accompanying option turns the condition on) are listed together in the sections below. In this case, the default value is listed first.

Basic Options

The options described below start and stop ECL, display on-line assistance, and perform other basic operations.

Option	*Meaning*
–HELP	Displays all of ECL's available options.
–INITIALIZE	Reinitializes the ECL settings to their default values. (The default values are listed earlier in this chapter in Table 2-1.) –INIT erases the current command history and clears both the search ring and the kill ring. –INIT also resets the internal prompts to use the default RDY set.
–OFF / –ON	Controls the command-line environment setting. –OFF resets the command-line environment to the standard PRIMOS terminal input routine; a message to this effect is displayed on your terminal screen. –ON sets the command-line environment to the ECL terminal interface. ECL displays an informational message when ECL is in effect. Note that –OFF and –ON affect only the use of ECL by PRIMOS, and that ECL retains its settings even when it is off.
–READY_BRIEF [system_prompt] **–ERROR_BRIEF [system_prompt]** **–WARNING_BRIEF [system_prompt]**	Specify the three categories of system prompt. All three options can be specified on one command line. –RB, –EB, and –WB each set their prompt (which default, respectively, to '`OK, `', '`ER! `' and '`OK, `'). Any replacements should be in single quotation marks that explicitly include the ending space. If **system_prompt** is not specified, the indicated prompt is set to the null string; this is useful when using the long-form PRIMOS RDY prompts. **system_prompt** may contain the placeholder character '**#**', which causes the current command history entry number to be displayed in its place. The placeholder may be escaped by specifying two number signs (**##**) in succession. See the discussion in the *PRIMOS User's Guide* for more information on prompt usage and specification.

ECL uses the standard PRIMOS RDY brief prompt set by default. ECL converts to its own internal set if any of the above prompt options are used to set an internal prompt. The # character is meaningful only in an internal prompt; it is treated as just another printable character within a RDY-specified prompt.

–VERSION — Displays ECL's full internal version and copyright notice, which are normally suppressed.

Terminal Bindings and Characteristics

The options described below help you define your terminal environment in ECL.

Option	*Meaning*
–BIND_TERM [bindings_file] [term_alias]	Specifies the file containing command bindings. This option is ignored if terminal bindings are successfully restored via the –RESTORE_TERM option. Otherwise, ECL first establishes the fundamental set of bindings, then reads the bindings file to supplement the fundamental set. Terminal-specific bindings are normally coupled with the terminal name provided by the –TERMINAL_TYPE option. You can override this setting by specifying a terminal name alias in **term_alias**, allowing slightly different capabilities or bindings to be combined. If no **bindings_file** is specified, the fundamental bindings are simply reestablished. If the –BIND_TERM option is not specified, current bindings are untouched, even if –TERMINAL_TYPE was specified. The format of the bindings file is discussed in Chapter 6. To disable bound sequences in the fundamental set, rebind them to the *unbound* command.
–BPS [b_rate]	Specifies the terminal baud rate. You can specify any standard baud rate between 110 and 19200; if you do not specify a baud rate, –BPS defaults to 9600. When you restore your terminal capabilities with the –RESTORE_TERM option, ECL ensures that the baud rate specified in the compiled bindings file matches that specified on the command line. The baud rate is used to properly compensate for the performance delays of certain terminal characteristics.
–HEIGHT [n]	Specifies the terminal height in **n** number of lines. This option overrides the height specified as **lines** in the TERMINFO database or the default height of 24 lines for the generic terminal. Omitting **n** or specifying **n** as 0 instructs ECL not to pause after every screenful of data when displaying the list of matches for the *expand_wild* command. (The *expand_wild* command is discussed in Chapter 6.)
–RESTORE_TERM [compiled_file]	Restores compiled terminal capabilities and bindings (including the current keyboard macro program, if any) from **compiled_file**. You can use this option and the –SAVE_TERM option on the same command line. If you do so, the **compiled_file** argument is optional for –SAVE_TERM; the compiled file becomes the one specified for –RESTORE_TERM. –RESTORE_TERM and –SAVE_TERM automatically work in conjunction with the –TERMINAL_TYPE,

–BPS, and –BIND_TERM options with no special intervention on your part. The –RESTORE_TERM functionality is discussed in detail in Chapter 6.

–SAVE_TERM [compiled_file]
Saves terminal capabilities and bindings (including the current keyboard macro program, if any) in a compiled format to **compiled_file**. You can use this option and the –RESTORE_TERM option on the same command line. If you do so, the **compiled_file** argument is optional for –SAVE_TERM; the compiled file becomes the one specified for –RESTORE_TERM. –RESTORE_TERM and –SAVE_TERM automatically work in conjunction with the –TERMINAL_TYPE, –BPS, and –BIND_TERM options with no special intervention on your part.

–SPEED [b_rate]
Synonymous with –BPS, above.

–TERMINAL_TYPE [type]
Specifies the terminal type of your terminal to ECL. This option is not case-sensitive. –BIND_TERM uses the terminal type specified by –TTP to set up a bindings file and –RESTORE_TERM verifies the terminal type when you restore a compiled bindings file. (These two options are discussed elsewhere in this section.) If you are not restoring or the restore operation is not successful, ECL searches the TERMINFO database for the entry **type** and initializes terminal capabilities. If **type** is not specified, ECL checks the global variable .TERMINAL_TYPE$ for a match; if it does not find one, ECL establishes capabilities for a generic 24-line by 80-column terminal.

–WALLPAPER [bindings_file
Displays the current command bindings, including the function key programs. If **bindings_file** is specified, ECL redirects the display to the specified file instead of the terminal. The display format conforms to the bindings file format discussed in Chapter 6. This means that files produced with this option can be reloaded using the –BIND_TERM option.

–WIDTH [n]
Specifies the terminal width in **n** columns. This option overrides the width specified as **cols** in the TERMINFO database or the default width of 80 columns for the generic terminal. Omitting **n** or specifying it as 0 tells ECL that you have a terminal incapable of wrapping.

–XOFF / –NO_XOFF
Interacts with the PRIMOS TERM command setting (–XOFF or –NO_XOFF), overriding it within the ECL command environment and restoring it within the PRIMOS command environment. –XOFF overrides any bindings to Ctrl S / Ctrl Q and uses them to stop and start output to the terminal, respectively. –XOFF is the default. –NO_XOFF enables Ctrl S / Ctrl Q to execute the ECL commands to whichthey are bound. ECL reverts to the original –XOFF duplex setting at command submittal. This means that you can start and stop terminal output with Ctrl S / Ctrl Q while a PRIMOS command is executing, even if you have specified –NO_XOFF. Once the command has finished executing, –NO_XOFF is again in effect. If your terminal must rely on Ctrl S / Ctrl Q for output control, then always specify –XOFF.

Controlling Terminal I/O

This section describes the options that control certain terminal I/O characteristics.

Option	*Meaning*
–CLEAN_COMO / –NO_CLEAN_COMO	Controls ECL terminal output that is captured by a command output (COMO) file. –CCOMO prevents the capture of all but the submitted commands and any ECL prompts displayed. –NCCOMO captures all character output within ECL. Be aware that –CLEAN_COMO only cleans COMO files opened by the local system. –CLEAN_COMO is the default.
–NO_CLEAR_LINE / –CLEAR_LINE	Controls ECL capability of initially clearing out the area on the terminal expected to be used for input. –CLEAR_LINE clears out the command line; NO_CLEAR_LINE does not. –NO_CLEAR_LINE is the default.
–ROW_MAJOR / –COL_MAJOR	Sets the display characteristics for automatic pathname completions. –ROW_MAJOR displays the list in alphabetical order within rows (as the PRIMOS LD command does). –COL_MAJOR orders the display alphabetically within columns. –ROW_MAJOR is the default, although –COL_MAJOR is more natural to read.
–NO_EDIT_COMI / –EDIT_COMI	Controls the editing of input from a command input (COMI) file. –NO_EDIT_COMI passes back each line of a COMI file as uninterpreted without recording it within the command history. –EDIT_COMI treats COMI file input as terminal input; all characters are passed through the ECL command interpreter and the submitted commands are incorporated into the command history. –NO_EDIT_COMI is the default.
–OBEY_ERKL / –NO_OBEY_ERKL	Controls the use of the PRIMOS erase and kill characters. –OBEY_ERKL always monitors and obeys the erase and kill characters as defined by PRIMOS, overriding any ECL binding associated with these characters. –OBEY_ERKL enforces the erase character as being bound to *rubout_char* and the kill character as being bound to *kill_region* (with a count of 4). –NO_OBEY_ERKL allows you to define your own erase and kill characters within ECL. –OBEY_ERKL is the default. For more information on the *rubout_char* and *kill_region* commands, see Chapter 6 and also see the *PRIMOS User's Guide*.
–SILENT	Instructs ECL to display only the most serious error messages, such as improper package installation or bad command-line syntax. –SILENT disables the display of most error, warning, or informational messages. This option is useful within CPL interfaces to ECL; in this case, the CPL program can examine the returned SEVERITY code and produce a message of its own.

Command History

This section describes the options that allow you to manipulate the command history.

Option	*Meaning*
–NO_STACK / –STACK	Controls the logical structure of the command history as either a ring or a stack. –NO_STACK specifies the structure as a ring. –STACK specifies the structure as a stack. In a ring structure, all submitted commands replace their ring entry; thus, changing a previous command replaces its original form. With a stack, each new command, even if it's only a modified previous command, is pushed onto the top of the stack and the history pointer is set to the top. While a stack minimizes lost commands, it is more difficult to execute a sequence of previous commands than with a ring. –NO_STACK is the default.
–NO_STICK / –STICK	Controls the positioning of the history pointer after a previous command has been recalled. –STICK does not reposition the history pointer; the pointer *sticks* to the position after the recalled command. –NO_STICK moves the history pointer back to the top of the stack (if you have used the –STACK option) or repositions the pointer after the last *new* ring entry (if you have used the –NO_STACK option). Making the history pointer stick allows a sequence of previous commands to be executed in a row. If you use –STICK within a ring history (by using the –NO_STACK option), subsequently entering a new command overwrites the corresponding hidden history command. The –SHOW_HIDDEN option discussed below informs you of this history command loss by revealing hidden commands automatically when used in conjunction with –STICK. –NO_STICK is the default.
–NO_SHOW_HIDDEN / –SHOW_HIDDEN	Controls the automatic display of the underlying hidden command in the command history at a new prompt (while the current event is within past history). –NO_SHOW_HIDDEN does not automatically reveal the hidden command, but instead provides a blank prompt line. –SHOW_HIDDEN reveals the hidden command at the new prompt and the cursor is placed at the *beginning* of this command. –SHOW_HIDDEN facilitates the rapid replay of previous consecutive commands in the command history without having to recall them explicitly by using, for example, the *prev_line* or *next_line* command. Also, the revealed command is automatically erased if the very next character sequence is not one intended to edit the hidden command (that is, when you start typing a new command). Note that –SHOW_HIDDEN is not useful if –NO_STICK is enabled because –NO_STICK prevents sticking within past history.
–RESTORE_HISTORY **history_file**	Restores the command history from **history_file**, which must be specified. The search ring and kill ring are restored along with the command history. Use this feature to save your command history (with the –SAVE_HISTORY option) just before logging out or reinitializing your command environment, and then to restore

the history upon logging back in. Note that ECL need not be enabled (with the –ON option) for these options to work, it needs only to be installed.

–SAVE_HISTORY history_file

Saves the command history to **history_file**, which must be specified (unless you use –SHIST on the same command line as –RHIST **history_file**). The search ring and kill ring are saved along with the command history. Use thisfeature to save your command history just before logging out or reinitializing your command environment, and then to restore the history (with the –RESTORE_HISTORY option) upon logging back in. Note that ECL need not be enabled (with the –ON option) for these options to work, it needs only to be installed.

Pathname Completion

This section describes the options that allow you to control automatic pathname completion and abbreviation expansion.

Option	*Meaning*
–NO_WILD_ABBREV / –WILD_ABBREV	Controls automatic abbreviation expansion as the first step in automatic pathname completion. (Automatic pathname completion is done with the *expand_wild* command.) –WILD_ABBREV, after extracting the partial pathname from the command line for completion, first references the PRIMOS abbreviation mechanism to expand any argument-type abbreviations that may exist within the specified pathname. Global variable references are also expanded. As in the PRIMOS command environment, expansion is not performed if the first character in the command line is a tilde (~). (Note that expansions are sensitive to cursor position within the pathname while the –WILD_TAIL option is in effect.) The default, –NO_WILD_ABBREV, disables automatic abbreviation expansion.
–WILD_DIRECTORY / –NO_WILD_DIRECTORY	Controls the automatic appending of the right-angle (>) character when the last element of an automatic pathname completion is a directory. The default, –WILD_DIRECTORY, causes > to always be appended. –NO_WILD_DIRECTORY specifies that you must explicitly type the right angle character when you want to reference a file system object within that directory. This option is useful because many PRIMOS commands, such as ATTACH, do not allow pathnames that end with the right-angle character. The default is –WILD_DIRECTORY.
–WILD_MENU / –NO_WILD_MENU	Controls the numbering of potential pathname completions displayed in the menu list generated by the *expand_wild* command. The default, –WILD_MENU, numbers each potential completion for selection by means of the *expand_wild_menu* command. –NO_WILD_MENU displays an unnumbered menu list.

–NO_WILD_TAIL / –WILD_TAIL	Controls wildcarding sensitivity to the cursor position within a partial pathname while you are using the automatic pathname completion facility. The default, –NO_WILD_TAIL, is not cursor sensitive and completes pathnames by wildcarding at the end of the partial pathname. –WILD_TAIL causes pathname completion by wildcarding at the cursor position within the partial pathname, and also includes any common matching tail substring after the cursor while inserting the results of an automatic completion.

Editing Characteristics

This section describes the options that allow you to control ECL's editing characteristics.

Option	*Meaning*
–NO_CASE_SEARCH / –CASE_SEARCH	Controls the case-sensitivity of the search commands *forward_search* and *reverse_search*. –NO_CASE_SEARCH instructs ECL to ignore cases when searching; –CASE_SEARCH indicates case-sensitivity when searching. –NO_CASE_SEARCH is the default.
–COMPONENT / –ENTRY	Establishes the definition of a word to ECL. This affects the various word commands such as *forward_word* and *rubout_word*. –COMPONENT, which is the default, indicates that a word consists only of alphanumeric characters and the underscore character (including letters from the ANSI International part of the Prime ECS character set). –ENTRY indicates that a word may contain any characters that are legal in a PRIMOS entryname, including wildcard (@) and name generation (=) characters. This supplements the alphanumeric characters and underscore with the following characters: `# $ & * / - . @ + ^ =`

Examples

In the first example, ECL is enabled for use at PRIMOS command level:

```
OK, ECL -ON
[EDIT_CMD_LINE Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.
Editing command input with ECL. (EDIT_CMD_LINE)
OK,
```

In the next example, a new RDY prompt is defined:

```
OK, ECL -RB '>> '
>>
```

In the next example, all of the RDY prompts are customized:

```
>> ECL -RB 'YES! ' -EB 'NO! ' -WB 'FLAG ' -ON
YES!
```

In the next example, the number-sign character (#) in the prompt substitutes the current command history entry number in place of the # character:

```
YES! ECL -RB '#: '
004:
```

In the next example, a common way to set up the command history is shown. This method allows you to recall a sequence of commands from the command history, but if you modify one of those commands, the modified command is added to the end of the history rather than replacing the original command in the history.

```
004: -ECL -STICK -SHOW_HIDDEN -STACK
005:
```

In the next example, the default bindings file is established, restored and saved on one command line. This is the recommended method for using a bindings file.

```
005: ECL -TTP PT200 -BTERM -RTERM COMPILED.FILE -STERM
Established PT200 terminal capabilities (9600 baud).
(EDIT_CMD_LINE)
Established default bindings (92% available). (EDIT_CMD_LINE)
Saved PT200 terminal capabilities and bindings (9600 baud).
(EDIT_CMD_LINE)
006:
```

The bindings file has now been established. In the next example, the same command line as above restores the default bindings.

```
006: ECL -TTP PT200 -BTERM -RTERM COMPILED.FILE -STERM
Restored PT200 terminal capabilities and bindings (9600 baud).
(EDIT_CMD_LINE)
007:
```

In the last example, all of the above options are enabled on the same command line. Use this method in your login routine to enable these options automatically.

```
007: ECL -ON -RB '#: ' -STICK -SHOW_HIDDEN STACK -TTP PT200
-BTERM -RTERM COMPILED.FILE -STERM
Editing command input with ECL. (EDIT_CMD_LINE)
Restored PT200 terminal capabilities and bindings (9600 baud).
(EDIT_CMD_LINE)
008:
```

▶ EDIT_PROFILE [pathname] [–PROJECT project-id]

EDIT_PROFILE invokes the PRIMOS profile editor.

The profile editor is used by the System Administrator (SA) or a Project Administrator (PA) to add, change, and delete information about user profiles and the projects to which users belong.

EDIT_PROFILE operates in three distinct modes:

- Project Administrator mode
- System Initialization mode
- System Administrator mode

Each of these modes has its own set of subcommands. Only the Project Administrator mode is discussed here because the other two modes can be used only by the System Administrator. See the *System Administrator's Guide, Volume III: System Access and Security* for an explanation of the two system modes.

User Profiles and Projects

A **user profile** is a list of attributes that identify a user ID. The attributes are summarized in Table 2-2.

TABLE 2-2
User Profiles

Item	*Modifiable*	*Comments*
User ID	SA	Mandatory for each user, but may be used by one or more users.
Login Password	SA or user	Passwords are usually unique. SA creates the initial password for a user; thereafter the user can change the password with the CHANGE_PASSWORD command. Blank or null passwords are allowed on some systems.
Initial Attach Point	SA or PA	Mandatory for each user. Attaches user to origin directory at login. The Initial Attach Point depends on the project (if any) specified by the user at login.

TABLE 2-2
User Profiles - Continued

Item	*Modifiable*	*Comments*
Project Affiliation	SA or PA	SA must assign one project to each user. The project may be a default project.
List of Access Groups	SA or PA	SA creates systemwide and/or project-specific access groups. SA assigns users to systemwide groups, which are active when a user logs in. PA may assign users to project-specific groups, which are active when a user logs in as a member of a particular project.

A **project** is an administrative grouping, such as a class, project team, or department. A project is not the same as an access group. See the SET_ACCESS command or the *PRIMOS User's Guide* for an explanation of access groups.

Each project is managed by a Project Administrator, who is appointed by the System Administrator. A Project Administrator can manage several projects at once. The Project Administrator mode of EDIT_PROFILE allows the Project Administrator to manage a project by performing the following functions:

- For existing project members, changing or listing their attributes as needed
- For the project as a whole, changing the project profile, adding new members to the list of project users, deleting users from that list, and maintaining the files that contain this information
- For managing more than one project, using EDIT_PROFILE options to effect changes on several projects at once

For a more detailed discussion of user profiles and projects, see the *System Administrator's Guide, Volume III: System Access and Security*.

Project Administrator Mode

A Project Administrator can change the attributes only of members of the particular project or projects that he or she administers.

To use EDIT_PROFILE in Project Administrator mode, you must specify the –PROJECT option with your project ID:

EDIT_PROFILE **[pathname] –PROJECT project-id**

Supply a pathname only when your project is not on your local system. The pathname is the name of the disk (partition) on which your project is kept. For example, if you are Project Administrator for a project called HARKNESS on a disk

called HAMPER, and you are logged in on another system, issue the following command:

```
OK, EDIT_PROFILE <HAMPER> -PROJECT HARKNESS
```

The project you specify becomes your current project. All subsequent subcommands are performed on the current project unless you specify an optional project ID in a subcommand. To change your current project, use the ATTACH_PROJECT subcommand.

EDIT_PROFILE Subcommands

The eleven EDIT_PROFILE subcommands of Project Administrator mode are described below. Enter the subcommands at the EDIT_PROFILE right angle-bracket (>) prompt.

▷ ADD_USER [user-id] [options]

ADD_USER adds a user to a project and creates the user's profile. If you specify only **user-id**, that user is added to your project as a new member with the default attributes described in the project profile. Use the –PROFILE option to establish a new profile.

Option	*Meaning*
–LIKE other-user-id	Assigns to a new user the same attributes as an existing user specified by **other-user-id**.
–NO_QUERY	Prevents the system from asking whether you want to check or change the profile of the new user.
–PROFILE	Creates a new user profile. The system prompts you for the attributes of the user. If you do not specify this option, the profile is set up from the default attributes in the project profile.
–PROJECT project-id	Names the project to which the user is added. Use this option only if you are adding the user to another project (other than the current project) that you administer.

▷ ATTACH_PROJECT [project-id]

ATTACH_PROJECT sets **project-id** as the new current project. If you do not specify **project-id**, you are prompted for it.

▷ CHANGE_PROJECT [project-id] [–PROFILE] [–LIST]

CHANGE_PROJECT changes the profile of the current project. You need not specify **project-id** unless you administer several projects.

The –PROFILE option causes the system to prompt you for the changes to the project profile. The –LIST option displays the current attributes of the project profile.

▷ CHANGE_USER [user-id [–PROJECT project-id] [–LIST]]

CHANGE_USER changes the profile of an individual member of your project. If you do not supply **user-id**, the system prompts you for it. The System Administrator can restrict the attributes that a Project Administrator can change.

Use the –PROJECT option if the user is a member of another project (other than the current project) that you administer. The –LIST option lists the user's attributes after the changes have been made.

▷ DELETE_USER [user-id [–PROJECT project-id]]

DELETE_USER deletes a user from the current project. Use the –PROJECT option only if the user is a member of another project that you administer.

▷ DETACH_PROJECT [project-id]

DETACH_PROJECT clears the setting of a current project set by a previous ATTACH_PROJECT or other EDIT_PROFILE subcommand. After you issue the DETACH_PROJECT command, you have no current project. If you give subsequent EDIT_PROFILE subcommands, you must either give the ATTACH_PROJECT subcommand first or use the –PROJECT option to specify to which project the subcommand applies.

▷ HELP [subcommand-name]

HELP displays arguments, options, and option arguments for the specified EDIT_PROFILE subcommand or for all EDIT_PROFILE subcommands if you do not specify a subcommand name.

▷ LIST_PROJECT [project-id] [options]

LIST_PROJECT displays a list of project and user attributes. The list always includes the project limits imposed by the System Administrator. You need not specify a project ID unless you want to list the attributes of another project (other than the current project) that you administer. The LIST_PROJECT options are

Option	*Meaning*
–ALL	Lists profiles for all users in the project.
–APPEND	Used with –OUTPUT to add the output to the end of the file, rather than overwriting it.
–OUTPUT pathname	Writes the output from the subcommand into the file specified by **pathname**. If you use only a simple filename, the file is placed in the SAD. This option is particularly useful if the –ALL option produces considerable output.
–PROFILE	Displays the project attributes.
–TTY	Displays output at the terminal. Because –TTY is the default, you should use it only with –OUTPUT to also display the output at the terminal.
–USER user-id	Displays the profile of the specified project member. Only one user can be specified.

▷ LIST_USER [user-id [{ –PROJECT project-id / –ALL }]]

LIST_USER displays the attributes of an individual member of your project. If you do not specify a user ID, the system prompts you for one.

Use the –PROJECT option if the user is a member of another project (other than the current project) that you administer. The –ALL option displays the user's attributes in each project that you administer.

▷ QUIT

QUIT leaves EDIT_PROFILE and returns you to PRIMOS command level.

▷ REBUILD [–PROJECT project-id] [–SIZE n]

REBUILD rebuilds your project to hold more members. Use the –PROJECT option only if you are rebuilding another project (other than the current project) that you administer. Use the –SIZE option to indicate the total number of members (specified by **n**) you want in the project. The total should include the number of new members you expect to add to the project. If you do not use –SIZE, EDIT_PROFILE determines the new project size, based on the current total of project members.

Project members cannot log in to your project while EDIT_PROFILE rebuilds it.

▶ ELIGTS tenths

Operator Command

ELIGTS modifies the eligibility timeslice for users. For details, see the *Operator's Guide to System Commands*.

▶ EMACS [pathname] [options]

EMACS invokes the EMACS character-oriented screen editor, a separately priced product. For detailed information, see the *EMACS Primer*, the *EMACS Reference Guide*, and the *EMACS Extension Writing Guide*.

▶ EXPAND_SEARCH_RULES

```
[ object_name [–LIST_NAME listname] ]
  –SUFFIX {suffix}
  –FILE
  –DIRECTORY
  –SEGMENT_DIRECTORY
  –ACCESS_CATEGORY
[ –REFERENCING_DIR {path_name}      ]
```

EXPAND_SEARCH_RULES provides the full pathname of a specified file system object or search list.

EXPAND_SEARCH_RULES uses the Search Rules facility to expand an objectname to a fully qualified pathname. **object_name** is used to specify the objectname to be expanded. **object_name** must include all suffixes. PRIMOS either returns the fully qualified pathname to the terminal screen, or issues a message indicating that the requested object could not be found. You can also invoke EXPAND_SEARCH_RULES as a CPL function. When invoked as a CPL function, EXPAND_SEARCH_RULES returns the fully qualified pathname to a variable in the CPL program.

EXPAND_SEARCH_RULES uses one of the user's search lists to determine the fully qualified pathname. You can specify the appropriate search list by using the LIST_NAME option. If you do not specify a LIST_NAME option, EXPAND_SEARCH_RULES selects the appropriate search list based on the suffix of the objectname.

PRIMOS selects the following search lists by default:

Suffix	*Search List*
.RUN	COMMAND$
.SAVE	COMMAND$
.CPL	COMMAND$
Other/no suffix	ATTACH$

EXPAND_SEARCH_RULES Options

The EXPAND_SEARCH_RULES options are as follows:

Option	*Description*
–ACCESS_CATEGORY	Specifies that the file system object sought is an access category. This option allows you to limit the search to access categories.
–DIRECTORY	Specifies that the file system object sought is a directory. This option allows you to limit the search to directories.
–FILE	Specifies that the file system object sought is a file. This option allows you to limit the search to files.
–LIST_NAME listname	Permits you to specify the name of the search list that PRIMOS should use to locate the named object. If you do not specify –LIST_NAME, PRIMOS selects the search list (ATTACH$ or COMMAND$) based on the suffix of the objectname. If you specify –LIST_NAME as COMMAND$, PRIMOS supplies suffixes to the objectname in the following sequence: .RUN, .SAVE, .CPL, no suffix.
–REFERENCING_DIR search_rule	Permits you to specify a search rule that PRIMOS substitutes for the REFERENCING_DIR entries in the search list. EXPAND_SEARCH_RULES uses this search rule to search for the file system object.
–SEGMENT_DIRECTORY	Specifies that the file system object sought is a segment directory. This option allows you to limit the search to segment directories.
–SUFFIX suffix	Specifies suffixes that PRIMOS appends to the objectname to conduct the search. The suffixes must begin with a period (for example, .RUN). You can specify a maximum

of eight suffixes following a –SUFFIX option. PRIMOS searches for suffixes in the order listed. If no match is found with all listed suffixes, PRIMOS searches for the object with no suffix.

For further information on the Search Rules facility, see the *Advanced Programmer's Guide, Volume II: File System.*

▶ F77 pathname [options]

F77 loads the Prime FORTRAN 77 compiler and compiles the object program from the ASCII file specified by **pathname**.

It is recommended that you give **pathname** an .F77 suffix. FORTRAN programs can be compiled in V mode or I mode but not in R mode. The F77 compiler can compile programs written in FORTRAN IV.

For detailed information on the operation and options of the F77 compiler, see the *FORTRAN 77 Reference Guide.*

▶ F77DML [[–INPUT] in-pathname] [[–OUTPUT] out-pathname] [[–ERROR] err-pathname] [–DYNAMIC] [–NO_LINE_NUMBER]

F77DML invokes the FORTRAN 77 Data Manipulation Language (DML) preprocessor.

The DML translates embedded DML statements in the source file specified by **in-pathname** into a form that is usable by the FORTRAN 77 compiler. The preprocessor automatically adds the suffix .F77 to the filename part of **in-pathname**. **out-pathname** is the name of the output file that contains the translated DML statements. **err-pathname** is the file that contains the error listing.

The output file is placed in the current directory, unless you use the –OUTPUT option to place it in another directory.

Use the –ERROR option to place the error listing file in a directory other than your current one.

–DYNAMIC allows programs to invoke a schema at runtime by a schema name rather than by the schema number obtained at compile time. The schema name is used at runtime to dynamically resolve the schema number.

–NO_LINE_NUMBER suppresses the generation of line numbers in the output file.

For further information, see the *DBMS Data Manipulation Language Reference Guide.*

▶ F77SUBS [in-pathname] [–OUTPUT out-pathname] [–LIST list-pathname]

F77SUBS invokes the FORTRAN 77 Subschema Data Definition Language (DDL) compiler.

in-pathname is a source file containing the subschema definition. The CBL Subschema DDL compiler transulates the source file into the subschema table. The subschema table is placed in the same directory as the schema table. Use the –OUTPUT option to place the subschema table in another directory.

The compiler also produces an output listing file that contains the source listing, error messages, and a map of the User Work Area. The file is called **in-pathname**.LIST. The listing file is placed in the directory that contains the schema table. Use the –LIST option to specify another directory for the listing file or a different name for the file.

For more information on F77SUBS, see the *DBMS Data Description Language Reference Guide.*

▶ FAP

FAP invokes the FORMS Administrative Processor.

The FORMS Administrative Processor creates and maintains the forms definition catalog, configures new terminals and new device drivers into the FORMS system, and obtains system status information. When invoked, FAP displays an asterisk (*) prompt and waits for you to enter a FAP command.

For detailed information on FAP, see the *FORMS Programmer's Guide.*

See also FDL; FED; FORMS.

▶ FAU

FAU invokes the PRISAM File Administrator Utility (FAU).

FAU is an interactive, menu-driven utility that manipulates PRISAM files. For details, see the *PRISAM User's Guide.*

▶ FDL pathname [options]

FDL invokes the FORMS Definition Language (FDL) compiler.

pathname is an FDL source file. It is recommended that you give **pathname** a .FORM suffix. For detailed information on the operation and options of FDL, see the *FORMS Programmer's Guide.*

See also FAP; FED; FORMS.

▶ FDML [in-pathname] [out-pathname] [err-pathname] [–INPUT in-pathname] [–OUTPUT out-pathname] [–ERROR err-pathname] [–DYNAMIC] [–NO_LINE_NUMBER]

FDML invokes the FORTRAN Data Manipulation Language (DML) preprocessor.

The DML preprocessor translates embedded DML statements in the source file specified by **in-pathname** into a form that can be used by the FORTRAN compiler. **out-pathname** is the name of the output file that contains the translated DML statements. The preprocessor automatically adds the suffix .FDML to **out-pathname**. **err-pathname** is the file that contains the error listing.

The output file is placed in the current directory, unless you use the –OUTPUT option to place it in another directory.

–DYNAMIC allows programs to invoke a schema at runtime by a schema name rather than by the schema number obtained at compile time. The schema name is used at runtime to dynamically resolve the schema number.

–NO_LINE_NUMBER suppresses the generation of line numbers in the output file.

For further information, see the *DBMS Data Manipulation Language Reference Guide*.

▶ FED [–PROFILE profile-pathname] [–CATALOG catalog-pathname]

FED invokes the FORMS Editor, FED.

Use FED to create and maintain form definitions. For further information, see the *FED User's Guide*.

See also FAP; FDL; FORMS.

▶ FILMEM [ALL]

FILMEM clears the contents of user address space to zeros for unsegmented (R-mode) programs.

Under PRIMOS, FILMEM with no argument fills memory locations 100_8 through 77777_8 with zeros. FILMEM ALL clears 100_8 through 177777_8.

▶ FILVER [pathname-1 pathname-2]

FILVER compares two files and reports any differences.

pathname-1 and **pathname-2** are the two files you want to compare. If you do not specify the pathnames, FILVER prompts you for them:

```
OK, filver

FILE 1: prog1
FILE 2: prog2
```

The two files are compared for equivalence. FILVER compares every halfword in **pathname-1** against the equivalent halfword in **pathname-2** (One halfword equals two bytes.) The differences are reported in a form useful for comparing runfiles. You should have listings of both files to make use of the comparison information printed by FILVER.

If both files are exactly the same, the following message is displayed:

```
FILES ARE EQUAL
```

If the files are not the same, each difference is displayed with the word DIFF followed by four six-digit numbers in octal:

DIFF w x y z

w and **x** describe the position of the file, **w** being a sector number and **x** being the offset within the sector. A sector is 1000_8 (512 decimal) halfwords long. (You must take into account the 18-byte header in runfiles and any offset from a sector boundary in the starting location of the runfile.) **y** is the value of the differing halfword in **pathname-1**; **z** is the value of the halfword in **pathname-2**.

If one file is longer than the other, FILVER also lists the lengths, in halfwords, of both files. (Halfwords are shown as "WORDS".)

If the FILVER output is more than one full screen, the prompt `CONTINUE=` is displayed. To display the next screen, enter YES, OK, or AYE in uppercase letters. To stop the output, enter NO or NAY in uppercase. Any other response produces a `WHAT?` error prompt.

The following example illustrates how FILVER operates. The files to be compared are named ALPHA and BETA.

```
OK, SLIST ALPHA
These are the times that try men's souls.
OK, SLIST BETA
These are the chimes that fry hen's soles.
OK, FILVER ALPHA BETA

DIFF 000000 000007 172351 161750
DIFF 000000 000010 166745 164755
DIFF 000000 000011 171640 162763
DIFF 000000 000012 172350 120364
DIFF 000000 000013 160764 164341
DIFF 000000 000014 120364 172240
DIFF 000000 000015 171371 163362
DIFF 000000 000016 120355 174640
DIFF 000000 000017 162756 164345
DIFF 000000 000020 123763 167247
DIFF 000000 000021 120363 171640
DIFF 000000 000022 167765 171757
DIFF 000000 000023 166363 166345
DIFF 000000 000024 127212 171656
FILE LENGTHS DIFFER AS FOLLOWS:
FILE 1: 21 WORDS
FILE 2: 22 WORDS

FILES NOT EQUAL
ER!
```

See also CMPF.

▶ FIND_RING_BREAK [{ -INPUT pathname / -HELP }]

Operator Command

FIND_RING_BREAK locates a break in a ring.

The FIND_RING_BREAK utility works only for hard breaks that cause complete interruption of the signals on the ring. **pathname** is the name of a file that contains the physical configuration of the ring. The –HELP option displays an online help screen that shows the syntax and options of the command.

For detailed information, see the *Operator's Guide to Prime Network Services.*

▶ FIX_DISK –DISK pdev [options]

Operator Command

FIX_DISK invokes a utility program that repairs disk partitions and maintains the integrity of a disk pack. Enter FIX_DISK –HELP for an online help screen. For detailed information, see the *Operator's Guide to System Commands* and the *Operator's Guide to File System Maintenance.*

▶ FSUBS in-pathname [–OUTPUT out-pathname] [–LIST list-pathname]

FSUBS invokes the FORTRAN Subschema Data Definition Language (DDL) compiler.

in-pathname is the name of the source file containing the subschema definition. The FORTRAN subschema compiler translates the subschema DDL into the subschema table. The subschema table is placed in the same directory as the schema table, unless you specify the –OUTPUT option to place the subschema table in another directory.

The compiler also produces a listing file (named **in-pathname**.LIST) that includes the DDL source test, error messages, and a map of the User Work Area. The listing file is placed in the directory that contains the schema table, unless you use the –LIST option to place it elsewhere.

If you enter FSUBS without arguments, you are prompted for the name of the source file:

```
OK, FSUBS
Source name?
:
```

For details, see the *DBMS Data Description Language Reference Guide* or the *DBMS Administrator's Guide.*

▶ FTGEN

Operator Command

FTGEN is the File Transfer Service (FTS) command for the System Administrator. FTGEN allows the administrator (logged in as SYSTEM) to configure the FTS system at a particular site, to initialize, and to validate the FTS database. The command also enables the administrator to display and modify the configuration. For detailed information on FTGEN, see the *PRIMENET Planning and Configuration Guide.*

See also FTOP; FTR; FTS.

▶ FTN pathname [options]

FTN loads the Prime FORTRAN IV compiler and compiles the object program from the ASCII file specified by **pathname**.

pathname should have the .FTN suffix, although this form is not mandatory. For detailed information on the options and operation of FTN, see the *FORTRAN Reference Guide*.

See also F77.

▶ FTOP [option]

Operator Command

FTOP is the File Transfer Service (FTS) operator command. FTOP provides special FTS operational privileges to the operator logged in as SYSTEM. With FTOP, the operator can initialize, monitor, and control the FTS servers. For detailed information on FTOP, see the *Operator's Guide to Prime Network Services*.

See also FTGEN; FTR; FTS.

▶ FTR { source_pathname [destination_pathname] [submit_options] / manage_option [request_name] }

FTR invokes the File Transfer Request (FTR) utility.

The File Transfer Request utility allows you to request and manage file transfers interactively between Prime computers connected by PRIMENET. FTR is part of the File Transfer Service (FTS), which is a separately priced product.

FTR offers two distinct functions: a transfer request function and a transfer management function. The first is used to submit a transfer request and the second to manage requests that have been submitted. Each function has its own FTR command-line format.

After you request a file transfer, FTR in turn submits the request to FTS, which then queues it for transfer. (Because FTS queues all requests on the local computer, you can make requests even when a communications link or remote computer is down.) After your request is queued, you can use the second FTR management function to display, modify, suspend, release, abort, or cancel the request.

Source and Destination Sites

File transfers take place between sites. A **site** is a single computer that is identified by a unique site name. (Prime sites normally use their PRIMENET node names as site names.) Files are transferred from a source site to a destination site. One of these sites must be your local site. The other site is usually a remote site, but it can be a different directory on the same local site. FTR cannot transfer files between two remote sites.

The following discussion assumes that you are using FTR between machines that have been configured using the FTGEN command. For information on transferring files to or from sites that are not configured, see the *User's Guide to Prime Network Services*.

Temporary Destination Files

Files are transferred initially to a temporary file to avoid accidental access to the destination file during the transfer process. The temporary file is renamed to the required destination filename upon successful completion of the transfer. For further details, see the *User's Guide to Prime Network Services*.

Request Names

Each request has two names associated with it: an external name and a unique internal name. You can use either name to identify the request. The **external name** is either the name of the file to be transferred or a specific name that you assign to the request when you submit it. You usually refer to the request by its external name. FTS assigns the **internal name**, always a number, which you may use to distinguish between two requests that have the same external name.

Requesting a File Transfer

To request the transfer of a file, use the following FTR command format:

FTR source-pathname [destination-pathname] [submit-options]

source-pathname specifies the name of the file to be transferred. If you specify a filename rather than a pathname, the file must be in the directory to which you are currently attached.

destination-pathname specifies the name the file is given at the destination site. Do not specify **destination-pathname** if you use the –DEVICE option.

submit-options, which are listed below, specify the source and destination sites and how the request is handled. You can specify two or more request submittal options if they do not conflict.

Submit Option	*Meaning*
–COPY	Transfers a copy of the file, not the original. (Default) This option does not affect files being transferred from a remote site.
–DEFER **date-time**	Specifies a date and time when the request is to be submitted for processing. If this option is not specified, the request is submitted immediately.
–DELETE	Deletes the local source file after it has been transferred successfully. (Default is –NO_DELETE only if you use –NO_COPY; otherwise, the copy is deleted.) This option does not affect files being transferred from a remote site.
–DEVICE LP	Transfers the file directly to the default line printer of a remote site. Only DAM and SAM files can be used with this option. Do not specify **destination-pathname** if you specify this option.
–DSTN_FILE_TYPE [type]	Specifies the destination file type. **type** is one of the following: SAM, DAM, SEGSAM, or SEGDAM. (Default is the source file type when sending and null when fetching.)
–DSTN_NTFY	Notifies the destination user when a file transfer starts and ends. Use –DSTN_USER to specify the destination user.
–DSTN_SITE **site-name**	Specifies the name of the site to which the file is being transferred. (Default is local site.) Required when sending a file to a remote site.
–DSTN_USER **user-name**	Specifies the owner of the file at the destination site. (Default is the login user ID for a local site and null for a remote site.)
–HOLD	Holds the request in a queue so that it is not initiated until you release the request with the –RELEASE request management option.
–LOG **pathname**	Specifies a log file that records the progress of the request, including whether the transferral was successful. If **pathname** exists, new log entries are appended to it; otherwise, a new log file is created.
–MESSAGE_LEVEL **level**	Specifies the level of detail entered in a log file. **level** is one of the following: NORMAL (default); DETAILED (logs all events); STATISTICS (same as DETAILED but with statistics); or TRACE (same as STATISTICS but with trace information). To use this option, you must also specify the –LOG option.
–NAME **external-name**	Specifies the external name of the request. (The default external name is the name of the source file.)

–NO_COPY	Transfers the original source file, not a copy of it.
–NO_DELETE	Does not delete the local source file after it has been transferred successfully. (Default)
–NO_DSTN_NTFY	Does not notify the destination user when a transfer starts and ends. (Default)
–NO_QUERY	Does not query you when you submit or modify a file transfer request. This option is useful when you are running FTS from a CPL file.
–NO_SRC_NTFY	Does not notify the source user when a transfer starts and ends. (Default)
–PRIORITY value	Sets the relative priority of a transfer request on the request queue. This means that you are no longer bound to the first in, first out queuing scheme that FTR uses by default. If you do not specify the –PRIORITY option, the default priority of a request is set by the Network Administrator using FTGEN.
–QUERY	Asks you to confirm any request questions. (Default)
–QUEUE queue-name	Specifies the name of the file transfer request queue in which a request is to be placed. (Default is the configured remote site queue, or OPEN_SYSTEM if that has been configured.) Normally, you use this option when sending a file to a remote site that has not been configured by FTGEN, in which case you must specify a queue that has been configured with FTGEN.
–SRC_FILE_TYPE [type]	Specifies the source file type. **type** is one of the following: DAM, SAM, SEGDAM, or SEGSAM. (Default is the type of the local source file; otherwise, the default is null.)
–SRC_NTFY	Notifies the source user when a transfer starts and ends.
–SRC_SITE site-name	Specifies the site from which the file is being transferred. (Default is the local site.) Required when transferring a file from a remote site.
–SRC_USER user-name	Specifies the owner of the file at the source site. (Default is the login user ID for a local source and null for a remote source.)

Sending a File

The basic FTR command format for sending a file from your local site to another Prime computer is

FTR source-pathname destination-pathname –DSTN_SITE sitename [submit-options]

submit-options are the options listed above.

The following example illustrates sending a file:

```
FTR JONES>NEWS SMITH>MAY.MEMO -DS SYS3 -LOG JONES>NEWS.LOG
```

This command sends a copy of the file NEWS from the directory JONES on your local system to the directory SMITH on system SYS3. The file is initially sent to the temporary file T$MAY.MEMO and is renamed to MAY.MEMO upon the successful completion of the transfer.

Fetching a File

The basic FTR command format for fetching a file from another Prime computer is

FTR **source-pathname destination-pathname** –**SRC_SITE sitename** **[submit-options]**

submit-options are the options listed above.

The following example illustrates fetching a file:

```
FTR BROWN>COMMENTS GREEN>NEWS -SS SYS3 -LOG GREEN>NEWS.LOG
```

In this example, a copy of the file COMMENTS from the directory BROWN on system SYS3 is transferred into the directory GREEN on your local system and is named NEWS. The log file is NEWS.LOG in the directory GREEN.

Managing Transfer Requests

To manage a request after it has been submitted, use the FTR command in the following format:

FTR **manage-option [request-name]**

manage-option specifies the type of action you want to take on the request. The options are listed below.

request-name identifies the file transfer request. You can use either an internal or external name as the request name. If you do not specify **request-name**, the action of the command is performed on all your requests.

The request management options, for which there are no abbreviations, are

Manage Option	*Meaning*
–ABORT **[request-name]**	Aborts a transfer in progress.
–CANCEL **[request-name]**	Cancels a file transfer request.
–DISPLAY **[request-name]**	Displays a detailed status of the request.
–HOLD **[request-name]**	Delays a file transfer until a user or operator releases the request with the –RELEASE option.

–MODIFY [request-name] [submit-options]	Changes the characteristics of a queued request. **submit-options** are the request submittal options listed above.
–RELEASE [request-name]	Releases a request held with the –HOLD option.
–STATUS [request-name]	Displays information about the current status of the request.
–STATUS_ALL	Displays information about all the requests on all the configured transfer queues.

Checking the Status of a Request

To check on the progress of requests that you have submitted, use the –STATUS option

```
FTR –STATUS [request-name]
```

request-name identifies the file transfer request. You can use either an internal or external name as the request name. If you do not specify **request-name**, the command produces a one-line summary on each of your waiting transfer requests.

FTR Help Screen

To display an online help screen listing the syntax and meanings of FTR options, issue the FTR command without an argument or issue the command

```
FTR –HELP USAGE
```

For complete information on FTR, see the *User's Guide to Prime Network Services*. *See also* FTGEN; FTOP; FTS.

FTS

FTS is the Prime communications File Transfer Service, a separately priced product that transfers files between computers connected by PRIMENET. FTS is not a command. The commands used with FTS are

- The FTR command, which provides the user interface to FTS
- The FTOP command, which provides operator control of FTS
- The FTGEN command, which allows the System Administrator to define the FTS environment

All three commands are discussed in this chapter. For full information on FTS, see the *User's Guide to Prime Network Services*.

▶ GENERATE_CATALOG –MT n [options]

GENERATE_CATALOG recreates a damaged ARCHIVE or BACKUP tape catalog, or verifies the catalog against its associated tape.

Only the System Administrator or operator can use this command with BACKUP tapes. Only the System Administrator, tape owner, or user with appropriate rights can use this command with ARCHIVE tapes. Before using GENERATE_CATALOG, you must assign a tape drive with the ASSIGN command and mount the correct tape.

To display an online help screen on GENERATE_CATALOG, use the command format

GENERATE_CATALOG –HELP [option]

For full information on the operation, arguments, and options of GENERATE_CATALOG, see the *Data Backup and Recovery Guide*.

▶ HDXSTAT

HDXSTAT displays the current status of all lines and sites of a half-duplex (HDX) network configuration.

When a network connection is defined as half duplex, communication between two Prime systems lasts only for the duration of the phone call.

The HDXSTAT display shows each defined HDX site, each defined HDX line, the association (where one exists) between individual sites and lines, the state of the telephone connection, and the status of the link. The link status is one of the following:

Link Status	*Meaning*
assigned	The line has been reserved for use by HDX PRIMENET, but is not yet set to receive or initiate calls.
awaiting call	A line has been set to receive a call from any remote site, but no remote site has yet called in.
not assigned	The line is not in use by HDX PRIMENET.
offline	No connection to the site exists.
running	The network connection to the remote site is working.
trying to establish	A line has been set to initiate calls to a remote site, but contact with the site has not been made.

For further information on HDXSTAT, see the *User's Guide to Prime Network Services*.

▶ HELP [name]

HELP displays online information about a command or a PRIMOS topic.

name is a PRIMOS command name (such as COPY or SPOOL) or a topic (such as COMPILERS or WILDCARDS). When you type HELP without a command or topic name, the command displays a list of commands and topics for which information is available.

If **name** is a command name, HELP displays the following: the command name; a one-line description of the command; the command abbreviation (if any); the command-line format; a summary of the function, options, and operation of the command; references to further information; and the date the HELP information was created or updated.

For example, to obtain information on the ATTACH command, enter the following command:

```
OK, HELP ATTACH
```

If **name** is a topic name, HELP displays information on that topic, including references to documentation. For example, to get information about Prime's debuggers, enter the following command:

```
OK, HELP DEBUGGERS
```

Similarly, for information on Prime technical publications, enter the command:

```
OK, HELP DOCUMENTS
```

This command displays a list of current manuals available from Prime and how to order them.

▶ HPSD

HPSD loads a version of PSD (the Prime Symbolic Debugger) that is stored in the upper portions of memory. For details, see the *Assembly Language Programmer's Guide*.

See also DBG; PSD; PSD20; VPSD; VPSD16.

▶ IDBMS [–CONFIG]

IDBMS initializes DBMS from the supervisor terminal.

When DBMS is shared, IDBMS executes automatically without the –CONFIG option. –CONFIG activates the CONFIG utility, which allows you to modify systemwide parameters that control

- The number of ROAM buffers allocated to a new transaction
- The length of time a transaction waits for the release of a read/write page lock
- The length of time a transaction waits for buffer allocation

New values for parameters become effective when you share DBMS or use IDBMS without the –CONFIG option. For further information, see the *DBMS Administrator's Guide*.

▶ INFO

INFO invokes the Prime INFORMATION system, a separately priced product. For detailed information, see the *INFO/BASIC Reference Guide*, the *Prime INFORMATION SIMPLE User's Guide*, the *PERFORM Reference Guide*, the *INFORM Reference Guide*, the *Prime INFORMATION Administrator's Guide*, the *INFORMATION Operator's Guide*, the *INFORMATION Configuration Guide*, and the *INFORMATION System Update* MRUs.

▶ INITIALIZE_COMMAND_ENVIRONMENT [–SERVER]

INITIALIZE_COMMAND_ENVIRONMENT resets a user's command environment to the state it was in at login.

Use INITIALIZE_COMMAND_ENVIRONMENT only when you suspect that your PRIMOS command environment has been damaged. Such damage is most clearly indicated if you receive the following display:

```
You have exceeded your maximum number of command levels.

You are now at mini-command level.  Only the commands shown below
are available.  Of these, RLS-ALL should return you to command
level 1.  If it does not, type ICE.  If this problem recurs,
contact your System Administrator.

Valid mini-commands are:

Abbrev  Full name                    Abbrev  Full name
- - -   - - - - - - - - - - - - - -   - - -   - - - - - - - - - - - - - - - -
C       CLOSE                        COMO    COMOUTPUT
DMSTK   DUMP_STACK                   ICE     INITIALIZE_COMMAND_ENVIRONMENT
LE      LIST_EPF                     LL      LIST_LIMITS
LMC     LIST_MINI_COMMANDS           LS      LIST_SEGMENT
        LOGIN                        LO      LOGOUT
P       PM                           PR      PRERR
        RDY                          REN     REENTER
RLS     RELEASE_LEVEL                REMEPF  REMOVE_EPF
S       START

OK,
```

Before using this command, you may solve the problem without reinitializing your environment by issuing the CLOSE –ALL command followed by the RELEASE_LEVEL –ALL command. If this does not work, then use INITIALIZE_COMMAND_ENVIRONMENT.

The INITIALIZE_COMMAND_ENVIRONMENT command performs the following operations:

- Deallocates any NPX slave processes active for this process
- Clears all open virtual circuits, except for the remote-login virtual circuit, if any
- Closes all open files, including a COMOUTPUT file
- Deallocates all your private dynamic and static segments
- Resets the Ring 3 command environment to an initial state
- Executes your login file
- Returns you to your origin directory (Initial Attach Point)
- Releases all assigned devices (except User 1)
- Resets terminal erase and kill characters
- Clears pending asynchronous write operations
- Releases RJE devices
- Closes named semaphores
- Enables messages created with the MESSAGE command
- Forces terminal output on

INITIALIZE_COMMAND_ENVIRONMENT is only one of several ways to escape from mini-command level.

–SERVER Option

When you use INITIALIZE_COMMAND_ENVIRONMENT with the –SERVER option, all the operations described above are performed. In addition, the –SERVER option performs the following operations:

- Terminates all of the server's InterServer Communication (ISC) sessions.
- Terminates all PRIMIX child processes that are a part of the server. The server is the server to which the caller's process belongs.
- Deletes all synchronizers and timers.

–SERVER is available only to a terminal or phantom process; it is not available to a child process.

For further information on INITIALIZE_COMMAND_ENVIRONMENT and on other ways to escape from the mini-command level, see the *PRIMOS User's Guide* and the *Programmer's Guide to BIND and EPFs*.

See also LIST_LIMITS.

▶ INPUT pathname

INPUT opens a source file on File Unit 1 for reading.

The file is assigned the name specified by **pathname**. If **pathname** is an filename, the file is placed in your current directory.

INPUT has the same effect as the command

OPEN pathname 1 1

For PMA and FTN, the source filename is usually provided with the command that starts assembly or compilation.

▶ IPSD

IPSD loads and initializes an extended version of the Prime Symbolic Debugger (PSD) that supports S-mode, R-mode, V-mode, and I-mode programs. Unlike the VPSD debugger, IPSD is able to map in the programs to be debugged, and can be used on both static-mode (SEG) programs and EPF programs.

For more information, see the *Assembly Language Programmer's Guide*.

See also DBG; HPSD; PSD; PSD20; VPSD; VPSD16.

▶ IROAM [–COLDSTART] *Operator Command*

IROAM is used by the ROAM Administrator to initialize ROAM and perform before-image recovery. IROAM can be issued only from the supervisor terminal. For details, see the *ROAM Administrator's Guide*.

▶ JOB { pathname [submit–options] | [job-id] manage-option | [job-id] monitor-option | job-number operator-option }

JOB submits and manages individual jobs run in the Batch subsystem.

JOB has four groups of options. The first group lets you submit a batch job; the second group lets you manage an active job; the third group lets you monitor one or all of your active jobs; and the fourth group lets the operator control users' active jobs.

Submitting a Job

Use the following format for submitting a batch job:

JOB pathname [submit-options]

pathname is either a CPL file with a .CPL suffix or a command input file. If you specify no options, the Batch monitor first searches for the CPL file named **pathname**. If no such file exists, the monitor looks in your working directory for the entryname part of **pathname** and assumes that it is a command input file.

If either file is found, the monitor submits the job by performing the following steps:

1. Assigns your login name to the job.
2. Places the job in an appropriate Batch queue.
3. Assigns the job a unique job number. The format of the job number is

   #qnnnn

   q is the number of the queue and **nnnn** is the job's number within the queue. #10125 is an example of a job number.
4. Adds the queue's default options and values.

Note

If you submit a job from a password directory, you must use the –HOME option with the password included in the pathname of the home directory and you must enclose the pathname in single quotation marks. For example,

```
JOB REPORTS>PRODUCE_REP -HOME 'RPDB SECREP'
```

Otherwise, the job aborts because the batch phantom cannot attach to the directory.

submit-options are one or more of the options listed below. All numbers given as arguments to options must be specified as decimal integers.

Submit Option	*Description*
–ACCOUNTING **text**	Specifies accounting information for the job. **text** cannot be longer than 80 characters and cannot be an explicit register setting (octal number) or be preceded by a hyphen without single quotation marks. If **text** contains spaces, commas, or comment designators (/*), the text must be enclosed within single quotation marks. The information is included in job displays, but is not used in running the job.
–ARGS **cpl-arguments**	Passes CPL arguments to the job being processed. JOB does not read the CPL arguments but only passes them to the CPL file when execution of the file begins. –ARGS must be the last option on a command line because everything that follows (except comments) is assumed to be the CPL arguments being passed.

–COMOUTPUT pathname

Opens a command output file (also called a COMO file) with the specified pathname at the beginning of job execution. If you specify an entryname rather than a pathname, the COMO file is opened in the directory to which you are attached at the time of job submittal. If you specify the –COMOUTPUT option for a job that itself issues a COMOUTPUT command, the file opened for the –COMOUTPUT option is closed when the COMOUTPUT command internal to the job is executed; if the two files share the same name, the file opened for the –COMOUTPUT option is overwritten.

–CPL

Runs **pathname** as a CPL file, regardless of its name. –CPL is not needed if **pathname** has a .CPL suffix or if you use the –ARGS option.

–CPTIME { **seconds** / NONE }

Specifies the maximum amount of CPU time (in seconds) allotted to the job. The job aborts if it exceeds the time limit. NONE places no time limit on the job. If the –CPTIME specification conflicts with the –CPTIME limit of a queue requested with the –QUEUE option, you are asked to specify a new limit. If you do not use the –QUEUE option and the specification conflicts with all queues, the error message **`No queue available for job`** is displayed.

–DEFER date

Defers job execution until the date and time specified. At job submittal, the acknowledgement message gives the date and time of deferral. If the specified date and time have already passed, you receive a warning message, and the job becomes eligible for immediate execution. If the system goes down and the specified date and time have passed when the system comes back up, the job becomes eligible for immediate execution. If a job is deferred until a time that falls outside its queue's active window, the job is executed at the beginning of the first active window after the specified date and time. For the format of **date**, see the section Wilcard Options in Chapter 4.

–ETIME { **minutes** / NONE }

Specifies (in minutes) the elapsed time allowed before the job is aborted. Details are the same as for –CPTIME.

–FUNIT number

Specifies the file unit used for command input. (Default depends on queue parameters but is usually 6.) Cannot be used for CPL jobs.

–HOME pathname

Specifies the directory in which a job executes. **pathname** cannot be either a null specification or a relative pathname (that is, the pathname cannot begin with *>) and cannot exceed 80 characters in length. –HOME has the same effect as providing an ATTACH command as the first line of the command file.

–NOTIFY

Notifies you at job termination that your job has either completed or aborted. If you previously specified MESSAGE –DEFER, notification is deferred until you return to PRIMOS command level. If you previously specified MESSAGE –REJECT or are not logged in when the job terminates, you do not receive notification.

–NO_COMOUTPUT

Cancels a request made with the –COMOUTPUT option for a COMO file. If no COMO file was requested, –NO_COMOUTPUT has no effect. The –NO_COMOUTPUT option usually is used with the –CHANGE option or to override a –COMOUTPUT option on the $$ command line.

–NO_DEFER

If you deferred job execution with the –DEFER option, –NO_DEFER makes the job eligible for immediate execution. If you did not request job deferral, –NO_DEFER has no effect. The –NO_DEFER option usually is used with the –CHANGE option or to override a –DEFER option on the $$ command line.

–NO_NOTIFY

Does not notify you at job termination. If you did not request notification with the –NOTIFY option, –NO_NOTIFY has no effect. The –NO_NOTIFY option usually is used with the –CHANGE option or to override a –NOTIFY option on the $$ command line.

–PRIORITY n

Sets the job's priority within its queue. The value of n ranges from 0 to 9, inclusive, with 9 being the highest (most favored) priority. (Default is queue-dependent.)

–PROJECT project_id

Runs a job under a project ID different from the project ID under which you logged in. A corollary to this feature is that you can issue the ORIGIN command within a batch job; the attach point is your Initial Attach Point in the project under which the job is running.

–QUEUE queuename

Places the job in a specific queue. (Use the BATGEN –DISPLAY command to obtain the queue names and characteristics.) (Default is the first queue available.) If a conflict results between –QUEUE and another option (for example, –CPTIME), you are asked to resubmit the job with different options.

–RESTART { YES / NO }

Determines if a job can be restarted after an abort or system shutdown. (Default is YES.)

Using the $$ Command to Submit a Job

You can specify most of the above options within the command input file itself by using the $$ command as the first noncomment line of the file. The command format is as follows:

$$ JOB { user-id / * } [submit-options]

user-id is the name under which you logged in. If you specify an asterisk (*) instead of a user ID, any user can submit the file to Batch. Otherwise, only a user logged in as **user-id** can submit it. **submit-options** are any of the job submittal options listed above, except for –ARGS and –CPL.

The following example shows a command line using the $$ command:

```
$$ JOB JONES -HOME JONES>REPORTS -CPTIME NONE -ETIME NONE
```

Any job can specify options with the JOB command, the $$ command, or both. If an option is specified twice, the JOB option overrules the $$ option. For example, if $$ called for –CPTIME 20 and JOB called for –CPTIME 10, the job would be allowed only 10 seconds of CPU time before being aborted.

Note

If **pathname** resides in or is executed from a password directory, you must use the –HOME option with the password included in the pathname and the pathname enclosed in single quotation marks:

JOB **pathname –HOME 'directoryname password'**

Otherwise, the job aborts because the Batch monitor cannot attach to the directory.

Managing an Active Job

After the job is in a Batch queue, use the following JOB command format to control the execution or characteristics of the job:

JOB **[job-id] manage-option**

job-id is either the job number or the job name.

The job number, described above in the section Submitting a Job, is the 5-digit number preceded by a number sign (#) that is assigned to a job when it is placed in a queue. Because each job number is unique, you should use the job number rather than the job name to identify a particular job.

The job name is the name of the file being run. If the job was submitted as a pathname (for example, JOB PRISM>COLOR>BLUE), its job name is the final element of the pathname (BLUE, in this example). More than one active job can

have the same job name if the same file was submitted more than once as a batch job. When managing a submitted job, you should use the job name only if you have one active job of that name.

manage-option is one of the options listed below. You can specify only one option at a time.

Manage Option	*Meaning*
–ABORT	Terminates the execution of a running job or cancels a waiting, deferred, or held job.
–CANCEL	Prevents the execution of a waiting, deferred, or held job. Does not halt the execution of a running job but does mark it as not restartable.
–CHANGE [submit–options]	Modifies the –ACCT, –ARGS, –COMOUTPUT, –CPTIME, –DEFER, –ETIME, –FUNIT, –HOME, –NOTIFY, –NO_COMOUTPUT, –NO_DEFER, –NO_NOTIFY, –PROJECT, or –RESTART options of a waiting job. The possible new values for each option are the same as those specified above in the section Submitting a Job. –CPL, –QUEUE, and –PRIORITY cannot be changed.
–RESTART	Aborts an executing job and then restarts the job if it is restartable. Normally used after a JOB –CHANGE command.

Modifying an Executing Job

To change the characteristics of an executing job, first use the JOB –CHANGE command and then the JOB –RESTART command, as in the following example:

```
JOB BLUE -CHANGE -DEFER 20:00 -NO_COMOUTPUT -NO_NOTIFY -RESTART YES
JOB BLUE -RESTART
```

In this example, the first command makes changes in the job BLUE and the second command aborts the job and restarts it with its new characteristics.

Note

Do not confuse the –RESTART YES/NO submit option with the –RESTART manage option. The first always takes YES or NO as an argument and determines whether the job can be restarted if it aborts. The second never takes an argument and actually aborts and restarts a changed job.

Monitoring an Active Job

Use the format below to monitor one or all of your active jobs:

JOB [job-id] monitor-option

job-id is either the job number or the job name, as described above in the section Managing an Active Job. If you do not specify **job-id**, information is displayed on each of your active jobs.

monitor-option is one of the options listed below. You can specify only one option at a time.

Monitor Option	*Meaning*
–DISPLAY	Displays the same information about a job as the –STATUS option, plus the home directory of the job and the values (both user-specified and queue defaults) for all JOB and $$ command options.
–STATUS	Displays the job's number, its execution state (waiting, deferred, held, or executing), its name, and the name of its queue.

The following example illustrates the –DISPLAY option:

```
OK, JOB -DISPLAY
[JOB Rev. 21.0 Copyright (c) Prime Computer, Inc. 1986]


Cpl job curly(#00029), user PANCHO deferred (queue normal-1).
Submitted today at 10:05:52 a.m., deferred until today at 8:00:00 p.m.
Priority=5, cpu limit=None, elapsed limit=None.
Project=FOOBAR, Notify=No, Como=<WWEST>PANCHO>TV>curly.como.
Home ufd=<WWEST>PANCHO>TV


Cpl job shemp(#10000), user PANCHO waiting (queue normal-2).
Submitted today at 9:03:14 a.m.
Priority=5, cpu limit=None, elapsed limit=None.
Project=DEFAULT, Notify=Yes.
Home ufd=<WWEST>PANCHO>TV


Cpl job moe(#30000), user PANCHO waiting (queue background-2).
Submitted today at 9:46:25 a.m.
Priority=5, cpu limit=None, elapsed limit=None.
Project=DEFAULT, Notify=Yes.
Home ufd=<WWEST>PANCHO>TV
OK,
```

The first line of the –DISPLAY option's output shows the job's name, number, user ID, execution state (waiting, deferred, held, or executing), and the name of its queue.

The second line shows the time of job submittal, the time of job initiation (if the job is currently executing), and the time of job deferral (if the –DEFER option was previously specified).

The third line shows the job's file unit number (unless the job is a CPL program), its priority within its queue, its CPU time limit, and its elapsed time limit.

The fourth line shows the job's default project ID or the project ID specified with the –PROJECT option, its notification status ("Yes" if the –NOTIFY option was specified; "No" otherwise), and the pathname of its command output file if the –COMOUTPUT option was specified.

The last line shows the job's home directory.

The following example illustrates the –STATUS option:

```
OK, JOB -STATUS
[JOB Rev. 21.0 Copyright (c) Prime Computer, Inc. 1985]

Job status listing for user PANCHO:

Jobid#   State            External  name          Queue
- - - -  - - - -  - - - - - - - - - - - - - - -   - - - - -
#00029 deferred  curly                            normal-1

#10000  waiting  shemp                            normal-2

#30000  waiting  moe                              background-2
OK,
```

JOB as an Operator Command

The operator uses JOB to control a waiting, deferred, or held job. The operator's form of the JOB command is

JOB **job-number operator-option**

job-number is the 5-digit number preceded by a number sign (#) that is assigned to a submitted job. **operator-option** is one of the following: –ABORT, –CANCEL, –HOLD, –RELEASE, or –RESTART.

The –HOLD option holds a waiting job in the queue. The job is kept active but is prevented from executing until the operator releases it with the –RELEASE option. These two commands are used for jobs that must execute at a particular time or that need peripheral equipment that is not available at present.

The operator –ABORT, –CANCEL, and –RESTART options are identical to the user options with those names.

For more information on the JOB command and batch jobs, see the *PRIMOS User's Guide*.

KBUILD

KBUILD builds MIDASPLUS files.

KBUILD is an interactive subsystem that builds MIDASPLUS data and index files from sequential disk files. When invoked, KBUILD asks you a series of questions about the file to be built.

KBUILD can process the following five types of files:

- ASCII text files (compressed)
- Binary files written from FORTRAN programs
- Binary files written by PRWS$
- COBOL-written text files (uncompressed)
- RPG-written files (uncompressed)

KBUILD is the easiest method of adding data to a new or existing MIDASPLUS file. For information on KBUILD and other methods of building MIDASPLUS data and index subfiles, see the *MIDASPLUS User's Guide*.

KIDDEL

KIDDEL deletes or cleans out MIDASPLUS files.

KIDDEL is an interactive MIDASPLUS utility that is used primarily to remove entries from the various parts of a MIDASPLUS file while retaining the template of the file for further use. KIDDEL can also delete the template entirely.

The KIDDEL options allow you to perform the following functions:

- Clean out entries in one or more secondary indexes.
- Clean out entries in all index and data subfiles.
- Delete an entire MIDASPLUS file.
- Delete one or more secondary index subfiles.

For detailed information on KIDDEL, see the *MIDASPLUS User's Guide*.

L

See LISTF; LISTING.

```
                  ┌ -LINE n [-TO num [-EXCEPT num-1 [...num-10]]] ┐   ┌ -ACTIVE      ┐
                  │ ┌ -NTSABF  ┐                                  │   │ -ALL         │
▶ LAB             │ │ -NTSBUF  │                                  │   │ -NO_HEADER   │
                  │ └ -REMBUF  ┘                                  │   │ -NO_WAIT     │
                  └ -HELP                                         ┘   └ -ON node_name ┘
```

LAB displays buffer sizes for asynchronous lines. It displays the current and initial buffer sizes for the user's own line or any lines that are assigned to the user.

For the network user, LAB displays buffer sizes for NTS lines and PRIMENET Remote login lines, as well as for terminal lines.

Use of the LAB command may be restricted by the DSM Administrator.

The LAB user options are as follows:

User Option	*Description*
-HELP	Displays the format of the command and a list of valid command line options. When you select -HELP all other options are ignored.
-ALL	Requests information on the configuration of your lines on the system. Do not specify -ALL with the -ACTIVE option.
-ACTIVE	Requests information on all lines currently assigned to you. Do not specify -ACTIVE with the -LINE or -ALL options.
-LINE n	Requests information on a single line, where **n** is the required decimal line number, or when used with the -TO option, the first line number in a series. Valid line numbers range from 0 to 512 for direct-connect and from 1024 to 1536 for NTS lines.
-TO num	Requests information on a range of consecutively numbered lines, where **num** specifies the last number in a series beginning at the line number **n** given in -LINE. The value of **num** ranges from 1 to 511 and must be greater than **n**. You can use -TO with the -EXCEPT option, if desired.
-EXCEPT num-1...num-10	Excludes up to ten lines from a range specified with the -TO option, where **num** is a valid decimal line nmber less than **m**. Separate each **num** with a single blank space in the command line.
-ON node_name	Allows a privileged DSM user to display the buffer sizes on a remote system, where **node_name** is the node name of the remote system. If the node name is omitted, the LAB command is executed on the local system.
-NO_HEADER	Supresses the output header.
-NO_WAIT	Supresses the **--More--** prompt.

LAB Example

The following example shows the use of the LAB –ALL option:

```
OK, LAB -ALL
[LAB Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

Node: NODE 1
+-------------------------------------------------------------------+
|       |       Current Buffers        ||   Default/Initial Buffers  |
|       |------------------------------++-----------------------------|
| Line  |  Input  | Output  |   DMQ    ||  Input  | Output  |   DMQ    |
|-------+---------+---------+----------++---------+---------+----------|
|  REM  |    2048,    2048,    ****    ||    2048,    2048,    ****    |
+-------------------------------------------------------------------+
```

LAB as an Operator Command

As an operator command, LAB must be issued at the supervisor terminal or by a DSM-privileged user. When issued by a DSM-privileged user, LAB displays buffer size information on all lines.

For more information see *System Administrator's Guide, Volume II: Communication Lines and Controllers*.

See also CAB.

▶ LABEL MTn [options]

LABEL reads or writes volume 1 (VOL1) level labels on magnetic tapes.

LABEL creates and reads the following types of VOL1 labels:

- ANSI X3.27-1978 or X3.27-1987 standard 9-track ASCII
- IBM 9-track EBCDIC
- IBM 7-track BCD
- Prime ANSI nonstandard 9-track ASCII

On read operations, LABEL displays the volume and owner IDs, creation date, access (ANS tapes only), and other information.

Before using LABEL, you must assign a tape unit with the ASSIGN command and mount the correct reel on the tape drive.

The –VOLID, –VOLSER, and –VOLUME options, which are interchangeable, allow LABEL to write a label to tape. If one of these is not present on the command line, LABEL performs a read operation. You must include one of these options on the command line if you are processing an unlabeled tape.

To display an online help screen for LABEL, use the command

```
LABEL -HELP
```

For a complete discussion of LABEL and tape labels, refer to the *Data Backup and Recovery Guide*.

▶ LAC

See LIST_ACCESS.

▶ LATE

LATE defers execution of the next command until a specified time.

After you issue LATE, you are asked to enter the time at which the next command is to be executed. The prompt is as follows:

```
Time of day (HHMM) to execute next command:
```

Enter the time in the format **HHMM**, where **HH** is the hour (00 through 23) and **MM** is the minute (00 through 59). LATE responds to this input with the following message:

```
Next command will be executed at HH:MM
```

Respond to this prompt by entering a command. The command is executed at the time you designated at the first prompt.

If you specify a time that is earlier than the current time, execution of the command is deferred until the following day. For example, if LATE is issued at 12:00 noon, the following is displayed:

```
Time of day (HHMM) to execute next command: 0230

Next command will be executed at 02:30 tomorrow.
```

If you include the colon, the first two digits designate hours and the last two digits designate minutes. If you omit the colon, the last two digits are considered minutes. For example, 30 is interpreted as 00 hours and 30 minutes. If no time specification is made, the default is 00:00 (midnight).

After you issue LATE and the next command, you cannot use your terminal until the specified time, unless you use CONTROL-P to abort from LATE. LATE is particularly useful for running batch files during periods when the system load is light.

▶ LD [pathname] [wild1...wild15] [options]

LD lists the contents of a directory and, optionally, the attributes of the entries.

pathname identifies the directory to be listed and contains the first wildcard name. The default pathname is your current directory and the default wildcard name is @@.

If you do not specify **pathname**, all entries in your current directory are listed because the LD command without arguments or options is equivalent to the command format

```
LD @@
```

wild1...wild15 means that the default @@ wildcard name can be replaced with a total of 15 wildcard names.

A directory entry is listed if it matches either the entry that is part of the pathname and/or one of the wildcard names. For example, the following command lists entries in the current directory whose names begin with SALE or end with .MEMO:

```
OK, LD SALE@@ @.MEMO
```

LD Output

By default, the LD command displays a header line followed by the directory's contents. The header line lists the following items:

- The directory's pathname
- Your access rights if you are in an ACL directory or your status (OWNER or NONOWNER) if you are in a password directory
- The number of records used by the directory and its files
- The quota for the directory
- At PRIMOS Revision 22.0, the word **robust**, if you are listing a directory on a robust partition. A **robust partition** is a new partition type, introduced at PRIMOS Revision 22.0. See the *Operator's Guide to File System Maintenance* for details.

The directory entries are sorted alphabetically within each entry type and listed four across, grouped in the following order: file, segment directory, directory, access category. Each type has a header that identifies the type and the number of entries in it.

An example of an LD output is shown below:

```
OK, LD

<PUBS>CARVER>PROJECT (ALL access) Robust
45 records in this directory, 47 total records out of quota of 0.

11 Files.

BOOK_SCHED          LOAD              LOGIN.ABBREV      LOGIN.CPL
LOGIN.LAST          NETENX.CPL        NETLINK.CPL       PROG.BIN
PROG.PASCAL         TOLL.BIN          TOLL.CBL

2 Segment Directories.

PROG.SEG            TOLL.SEG

2 Directories.

MAIL                PERSONAL

1 Access category.

GUARD.ACAT

OK,
```

Use one or more of the LD options to produce a different output.

If the LD output is over 23 lines, the following prompt is displayed:

```
--More--
```

To abort the output, enter Q, QUIT, N, or NO in uppercase or lowercase letters. To continue the listing, press RETURN or enter any other characters.

LD Options

The LD options, listed below, can be specifed in any order.

Option	*Meaning*
–BRIEF	Limits the top header to a one-line format. Also eliminates the column labels for options (such as –DETAIL or –DTM) that normally produce them.
–CATEGORY_PROTECTED **[categoryname]**	Lists only the entries that are protected by the access category identified by **categoryname**. If you do not specify **categoryname**, all entries protected by any access categories are listed.

-DEFAULT_PROTECTED

Lists only the entries protected by default access control.

-DETAIL

Lists all attributes for each entry you select. (Default lists only the entrynames.) The attributes are displayed on three lines following the entryname.

On the first line:

1. Name of entry.
2. Date and time entry was last modified.
3. Date and time entry was backed up to tape by the BACKUP command (displays `** dtb not set **` if the entry has not yet been backed up or if BACKUP is not used on your system).
4. Date and time entry was created (displays `** dtc not set **` for entries with undefined DTC values, for example, entries in pre-Rev. 20.0 directories).

On the second line:

1. Date and time entry was last accessed (displays `** dta not set **` for entries with undefined DTA values, for example, entries in pre-Rev. 20.0 directories).
2. Size of entry in physical disk records (files, directories, and segment directories only). A size of -1 is shown for open COMO files and for entries to which you do not have Read (R) and List (L) access if the read/write concurrency lock setting is either SYS or EXCL; the actual record size is shown for open COMO files and for entries to which you do not have Read (R) and List (L) access if the read/write concurrency lock setting is either UPDT or NONE (see below).
3. Read/write concurrency lock setting, which specifies the number of users who can access the entry at the same time (files and segment directories only). The setting is one of the following: SYS (for system default); EXCL (for any number of readers or one writer); UPDT (for any number of readers and one writer); or NONE (for any number of readers and writers).
4. Quota of entry in physical disk records (directories only).
5. Dump switch indicator ("dmp" indicates entry has been dumped, and "nodmp" indicates entry has not been dumped; for files, directories, and segment directories only).

6. Access rights available to you; corresponding protection keys are listed for password-protected entries.
7. Delete-protection switch ("dprot" indicates the delete-protection switch was set with the SET_DELETE command; for files, directories, and segment directories only).

On the third line:

1. Type of entry ("sam," "dam," or "cam" for files; "sseg" or "dseg" for segment directories; "dir" for directories; "acat" for access categories).
2. RBF indicator ("RBF" means the file is a Recovery Based File, and a blank means it is not).
3. Truncate indicator ("trunc" means the entry was truncated by a FIX_DISK operation, and a blank means it was not; for files only).
4. Name of access category protecting the entry ("Specific ACL" indicates protection by a specific ACL, and "Default ACL" indicates default protection; for files, directories, and segment directories only).

–DTA Lists each entry's type and the date and time the entry was last accessed. If an entry has an undefined DTA value, displays `** dta not set **.`

–DTB Lists each entry's type and the date and time the entry was last backed up. If an entry has not been backed up, displays `** dtb not set **.`

–DTC Lists each entry's type and the date and time the entry was created. If an entry has an undefined DTC value, displays `** dtc not set **.`

–DTM Lists each entry's type and the date and time the entry was last modified.

–HELP Displays LD's syntax and options. The –HELP usage list is also displayed if PRIMOS detects an error while parsing the command.

–NO_COLUMN_HEADERS Suppresses column labels.

–NO_HEADER Suppresses the top header line and entry type headers. Also suppresses the type totals if specified with any sorting option. This option is most useful when combined with the –SINGLE_COLUMN option.

–NO_SORT Does not sort entries by name or by type and does not display type headers. This option takes precedence over any other sorting option specified on the command line.

–NO_WAIT	Suppresses the `--More--` prompt and does not pause after every 23 lines of output, but scrolls the output continuously.
–PROTECT	Lists the protection information for each entry. Protection information includes the access rights, the delete-protection, and the type of protection.
–REVERSE	Reverses the sort order from its default, but does not affect the sort order of entry types.
–SINGLE_COLUMN	Lists each entry on a separate line. Use this option only with the default output format in which only names are listed.
–SIZE	Lists the size of each entry and the quota for directories only. Displays a size of -1 for entries to which you do not have Read (R) access and List (L) access.
–SORT_DTA	Sorts entries within their type by descending date/time last accessed information and lists the date and time each entry was last accessed. If you try to sort by DTA in a pre-Rev. 20.0 directory, no sorting occurs, and each object returns `** dta not set **`. When only some of the objects you are sorting have undefined DTA values, these objects appear at the end of the list. Cannot be used with any other sorting option.
–SORT_DTB	Sorts entries within their type by descending date/time backed-up information, and lists the date and time each entry was last backed up. If an entry has not been backed up, displays `** dtb not set **`. Cannot be used with any other sorting option.
–SORT_DTC	Sorts entries within their type by descending date/time created information and lists the date and time each entry was created. If you try to sort by DTC in a pre-Rev. 20.0 directory, no sorting occurs, and each object returns `** dtc not set **`. When only some of the objects you are sorting have undefined DTC values, these objects appear at the end of the list. Cannot be used with any other sorting option.
–SORT_DTM	Sorts entries within their type by descending date/time modified information, and lists the date and time each entry was last modified. Cannot be used with any other sorting option.
–SORT_NAME	Sorts entries alphabetically by name only, not within their type. Does not display type headers. Cannot be used with any other sorting option.
–SORT_SIZE	Sorts entries by descending size within each type, and lists sizes of entries (except for access categories) and quotas of directory entries. Cannot be used with any other sorting option.

–SPECIFIC_PROTECTED	Lists only the entries protected by specific ACLs.
–WIDE	Lists output in lines wider than the 79-character line default. Uses 100-character lines for name-only output; otherwise, assumes name and all attributes fit on one line. This option is useful for output to printers that can handle wide paper.

Note

Using the –DETAIL option or the –SIZE option on an entry in a Rev. 20.0 or later directory updates the parent directory's DTA, not the entry's own DTA. However, using either the –DETAIL or the –SIZE option from a pre-Rev. 20.0 system on an entry in a Rev. 20.0 or later directory updates the DTA values of both the object and its parent directory.

You can also use any wildcard option (except for –VERIFY and –NO_VERIFY) with LD, even if you do not specify a wildcard name. For example, LD –FILE lists only the files in your current directory and LD –DIR lists only the directories. The wildcard options can be used in conjunction with the LD options (for example, LD –FILE –SORT_SIZE). See Chapter 4 of this guide for details on wildcard names and wildcard options.

Interrupting the LD Command

If you use LD when you are attached to a remote partition and the network experiences a momentary problem, PRIMOS may lose your attach point. In this case, the output of the LD command is as follows:

```
OK, LD
Virtual circuit was cleared. (current_directory) (ld)
ER!
```

To correct this problem, use the ATTACH command to reattach to the original parition, or to another parition.

Similarly, when you issue LD to list a directory to which you are not currently attached, PRIMOS temporarily attaches you to the directory that you are listing. If you allow LD to complete without interruption, your attach point is returned to the directory you were in when you issued the LD command (your current attach point). If you issue CONTROL-P (QUIT) after issuing LD, but before LD completes, the LD command may be interrupted before PRIMOS has a chance to return you to your current attach point. Type S to allow the LD command to finish and to return to your current attach point. Otherwise you may have to reset your attach point yourself.

LD Examples

The following series of examples illustrate the LD command.

Example 1: Listing the date and time each object in the current directory was last accessed:

```
OK, LD -DTA

<PUBS3>EDWARD (ALL access)
46 records in this directory, 48 total records out of quota of 0.

8 Files.
name              date/time accessed  type  rbf
------------------------------------------------------------------------------
LOGIN.ABBREV      15 Jan 88 10:15:56  sam
LOGIN.CPL         15 Jan 88 10:15:56  sam
TEST1             15 Jan 88 10:19:20  dam
TEST2             14 Jan 88 12:03:04  cam
TEST3             14 Jan 88 12:05:12  sam
TEST4             14 Jan 88 12:05:20  dam
TEST5             15 Jan 88 10:19:20  cam
WHERE.CPL         14 Jan 88 11:43:44  sam

2 Directories.
name              date/time accessed  type  rbf
------------------------------------------------------------------------------
OUTPUT            15 Jan 88 10:17:16  dir
RESULTS           15 Jan 88 10:17:08  dir

1 Access category.
name              date/time accessed  type  rbf
------------------------------------------------------------------------------
NO.TOUCH.ACAT     15 Jan 88 10:18:24  acat

OK,
```

Example 2: Listing the date and time each object beginning with TEST (using wildcard tokens) was created:

```
OK, LD TEST@@ -DTC

<PUBS3>EDWARD (ALL access)
46 records in this directory, 48 total records out of quota of 0.

5 Files.
name              date/time created   type  rbf
------------------------------------------------------------------------------
TEST1             14 Jan 88 11:50:44  dam
TEST2             14 Jan 88 12:03:04  cam
TEST3             14 Jan 88 12:05:12  sam
TEST4             14 Jan 88 12:05:20  dam
TEST5             15 Jan 88 10:19:16  cam

OK,
```

Example 3: Sorting all objects in the current directory by the date and time each was last accessed:

```
OK, LD -SORT_DTA

<PUBS3>EDWARD (ALL access)
46 records in this directory, 48 total records out of quota of 0.

8 Files.
name              date/time accessed  type  rbf
------------------------------------------------------------------------------
TEST1             15 Jan 88 10:19:20  dam
TEST5             15 Jan 88 10:19:20  cam
LOGIN.CPL         15 Jan 88 10:15:56  sam
LOGIN.ABBREV      15 Jan 88 10:15:56  sam
TEST4             14 Jan 88 12:05:20  dam
TEST3             14 Jan 88 12:05:12  sam
TEST2             14 Jan 88 12:03:04  cam
WHERE.CPL         14 Jan 88 11:43:44  sam

2 Directories.
name              date/time accessed  type  rbf
------------------------------------------------------------------------------
OUTPUT            15 Jan 88 10:17:16  dir
RESULTS           15 Jan 88 10:17:08  dir

1 Access category.
name              date/time accessed  type  rbf
------------------------------------------------------------------------------
NO.TOUCH.ACAT     15 Jan 88 10:18:24  acat

OK,
```

Example 4: Sorting all objects beginning with TEST (and wildcard tokens) by the date and time each was created:

```
OK, LD TEST@@ -SORT_DTC

<PUBS3>EDWARD (ALL access)
46 records in this directory, 48 total records out of quota of 0.

5 Files.
name              date/time created   type  rbf
------------------------------------------------------------------------------
TEST5             15 Jan 88 10:19:16  cam
TEST4             14 Jan 88 12:05:20  dam
TEST3             14 Jan 88 12:05:12  sam
TEST2             14 Jan 88 12:03:04  cam
TEST1             14 Jan 88 11:50:44  dam

OK,
```

Example 5: Listing all objects in the current directory with the –DETAIL option and suppressing the `--More--` prompt with the –NO_WAIT option:

```
OK, LD -DETAIL -NO_WAIT

<PUBS3>EDWARD (ALL access)
46 records in this directory, 48 total records out of quota of 0.

8 Files.
name              date/time modified  date/time backedup  date/time created
                  date/time accessed     size   rwlock    dump   access  delprot
                  type  rbf   trunc  protected by
------------------------------------------------------------------------------
LOGIN.ABBREV      14 Jan 88 11:52:56  ** dtb not set **   16 Dec 87 17:51:00
                  15 Jan 88 10:15:56        2       sys   nodmp       ALL
                  sam                (Default ACL)
LOGIN.CPL         16 Jun 87 16:37:32  ** dtb not set **   16 Dec 87 17:51:00
                  15 Jan 88 10:15:56        1       sys   nodmp       ALL
                  sam                (Default ACL)
TEST1             14 Jan 88 11:50:44  ** dtb not set **   14 Jan 88 11:50:44
                  15 Jan 88 10:19:20        7       sys   nodmp       ALL
                  dam                (Default ACL)
TEST2             14 Jan 88 12:03:04  ** dtb not set **   14 Jan 88 12:03:04
                  14 Jan 88 12:03:04        7       sys   nodmp       ALL
                  cam                (Default ACL)
TEST3             14 Jan 88 12:05:12  ** dtb not set **   14 Jan 88 12:05:12
                  14 Jan 88 12:05:12        7       sys   nodmp       ALL
                  sam                (Default ACL)
name              date/time modified  date/time backedup  date/time created
                  date/time accessed     size   rwlock    dump   access  delprot
                  type  rbf   trunc  protected by
------------------------------------------------------------------------------
TEST4             14 Jan 88 12:05:20  ** dtb not set **   14 Jan 88 12:05:20
                  14 Jan 88 12:05:20        7       sys   nodmp       ALL
                  dam                (Default ACL)
TEST5             15 Jan 88 10:19:20  ** dtb not set **   15 Jan 88 10:19:16
                  15 Jan 88 10:19:20        7       sys   nodmp       ALL
                  cam                (Default ACL)
WHERE.CPL         16 Jun 87 16:39:16  ** dtb not set **   16 Dec 87 17:51:00
                  14 Jan 88 11:43:44        1       sys   nodmp       ALL
                  sam                (Default ACL)

2 Directories.
name              date/time modified  date/time backedup  date/time created
                  date/time accessed     size    quota    dump   access  delprot
                  type  rbf   trunc  protected by
------------------------------------------------------------------------------
OUTPUT            15 Jan 88 10:17:16  ** dtb not set **   15 Jan 88 10:17:16
                  15 Jan 88 10:17:16        1         0   nodmp       ALL
                  dir                (Default ACL)

name              date/time modified  date/time backedup  date/time created
                  date/time accessed     size    quota    dump   access  delprot
                  type  rbf   trunc  protected by
------------------------------------------------------------------------------
RESULTS           15 Jan 88 10:17:08  ** dtb not set **   15 Jan 88 10:17:08
                  15 Jan 88 10:17:08        1         0   nodmp       ALL
                  dir                (Default ACL)
```

```
1 Access category.
name              date/time modified  date/time backedup  date/time created
                  date/time accessed   access  type  rbf
-------------------------------------------------------------------------------
NO.TOUCH.ACAT     15 Jan 88 10:18:24  ** dtb not set **   15 Jan 88 10:18:24
                  15 Jan 88 10:18:24      ALL  acat

OK,
```

▶ LE

See LIST_EPF.

▶ LEM { segdir-pathname>subfile-number / filename } [–NO_WAIT]

LEM lists the contents of the extent map of a Contiguous Access Method (CAM) file, on either Prime standard partitions or robust partitions.

segdir-pathname>subfile-number is the segment directory pathname and subfile-number of the desired CAM segment directory subfile.

filename is the name of the CAM data file.

–NO_WAIT suppresses the `--More--` prompt and allows continuous scrolling.

The extent map occupies one disk record and is the first record in a CAM file. It serves as an index to all other records in the file. The extent map records the number of extents in the file, the length and location of each extent, the extent length, and the physical and logical end of file. In a CAM file, the physical size is the number of disk records allocated to the file and the logical size is the number of disk records that contain data.

For more information on CAM files, see the *Operator's Guide to File System Maintenance* and *Subroutines Reference II: File System.*

LEM Output

The following example shows the extent map for subfile 2. Note that the term *record* (or *disk record*) represents 2048 bytes.

```
OK, lem master.cam.dbms>2
[LEM Rev 22.0]

Pathname: <SYS1>JOAN.Q>MASTER.CAM.DBMS>2.
Logical end of file:        113 (records)  115712 (half-words).
Physical end of file:       118 (records).
Minimum extent length:       16 (records).
```

```
Composed of 3 extents.

   Extent      Address        Length
               (octal)        (dec)
   ******      *********      *******
     1            537           50
     2            604           52
     3            726           16
```

After displaying the file's pathname, the above output shows the extent map's header followed by information about each extent. A description of each line in the output follows.

The four lines of the header are described below.

Line	*Explanation*
Logical end of file	Offset of (distance to) the logical end of file in half-words. In SAM and DAM files the logical end of file record equals the last physical record in the file; the logical end of a CAM file refers to where the data ends and not to empty disk records allocated to the file.
Physical end of file	Offset of (distance to) the physical end of file in records, including all disk records allocated to the file, whether used or unused.
Minimum extent length	Minimum number of disk records required to be allocated when additional space is needed for the file. The default minimum is 0.
Composed of n extents	Number of extents occupied by this file or subfile.

The second half of the display shows information about each extent. For each extent, this display contains the three columns described below.

Column	*Explanation*
Extent	Extent number. This number is not physically recorded in the table; in displaying the table's contents, the system automatically assigns a number for each extent entry according to its position in the table.
Address	The address of the first block in the extent. This address is an octal number.
Length	The number of blocks in the extent. This is a decimal number.

See also LIST_CONTIGUOUS_BLOCKS.

▶ LG

See LIST_GROUP.

▶ LISTF

LISTF displays information on the current directory.

The LISTF default output displays a header followed by the names of the files, access categories, subdirectories, and segment directories in the current directory. The display does not list each object's type. The information in the header is the same as for the LD command.

LISTF is now obsolete and has been replaced by the LD command.

▶ LISTING pathname

LISTING opens a file for writing on File Unit 2.

A file opened by LISTING is usually an output file for a compiler or assembler. All subsequent compilation and assembly listings are written to this file until it is closed. **pathname** is the name of the listing file. If **pathname** is a simple filename, the file must be in your current directory. LISTING has the same effect as the following command:

```
OPEN pathname 2 2
```

▶ LIST_ACCESS [pathname]

LIST_ACCESS lists the access rights connected to a file system object.

pathname is the file, directory, segment directory, or access category whose rights you want to list. If **pathname** is not specified, the access rights for your current directory are listed, as in the following example:

```
OK, LIST_ACCESS

ACL protecting  "<Current directory>"
        ALEX:      ALL
        .LEAD:     ALL
        $REST:     LUR
OK,
```

In this example, the user with ID ALEX and members of the access group .LEAD have ALL rights to the current directory, while all other users have List, Use, and Read rights.

If **pathname** is a file, directory, or segment directory, the command lists the ACL that protects the object and the priority ACL (if any) on the disk. The following example lists the rights of subdirectory STANZA, which is subordinate to the current directory POEM:

```
OK, LIST_ACCESS <KEATS>POEM>STANZA

"<KEATS>POEM>STANZA" protected by default ACL (from "<KEATS>POEM"):
        JOHN:   ALL
        $REST:  LUR
OK,
```

If **pathname** is an access category, LIST_ACCESS displays its contents, the Access Control List (ACL). You do not have to include the access category's .ACAT suffix when specifying its name, unless its directory also has an entry whose entryname is the same as the unsuffixed name of the access category. For example, use the following command to list the contents of the access category TOP.ACAT in your current directory:

```
OK, LIST_ACCESS TOP
```

The contents of TOP.ACAT are listed, unless you have a file called TOP in your current directory, in which case the contents of the ACL protecting TOP are displayed.

Note

If the access rights on a default-protected directory are changed while you are attached to that directory, LIST_ACCESS does not display the new access rights to you until you reattach to that directory. If the access rights on a directory protected by a specific ACL are changed while you are attached to that directory, LIST_ACCESS immediately displays the new access rights to you.

For more information on ACLs, access categories, and the meanings of access rights, see the SET_ACCESS command or the *PRIMOS User's Guide*.

See also EDIT_ACCESS; SET_ACCESS.

▶ LIST_ASSIGNED_DEVICES [options]

Operator Command

LIST_ASSIGNED_DEVICES displays all the devices on the system that have been assigned with the PRIMOS ASSIGN command. LIST_ASSIGNED_DEVICES is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_ASYNC [options]

Operator Command

LIST_ASYNC displays the status and configuration of any or all of the system's asynchronous lines. LIST_ASYNC is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_CATALOG [pathname] [options]

LIST_CATALOG lists the contents of catalogs created by the ARCHIVE and BACKUP commands.

By default, LIST_CATALOG lists the contents of ARCHIVE tape catalogs in the subdirectory CATS* which in located in the user's origin directory. If CATS* is located elsewhere, its pathname must be specified. Only the tape owner or System Administrator can list an ARCHIVE catalog.

BACKUP tape catalogs are maintained in the system catalog directory BACKUP*>CATS* and are listed with the –BACKUP option.

LIST_CATALOG supports the iteration, wildcarding, and treewalking command-line features. To display online help information on LIST_CATALOG, use the command

```
LIST_CATALOG -HELP
```

For details on LIST_CATALOG and its options, see the *Data Backup and Recovery Guide*.

▶ LIST_COMM_CONTROLLERS [options]

Operator Command

LIST_COMM_CONTROLLERS displays information on all the communication controllers present in a system, excluding the Prime Node Controller (PNC). LIST_COMM_CONTROLLERS is used with the Distributed Systems Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_CONFIG [directive_names] [options]

Operator Command

LIST_CONFIG displays the cold-start values, default values, and current values of the system variables that can be set by configuration directives at cold start. LIST_CONFIG is used with the Distributed Systems Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_CONTIGUOUS_BLOCKS [partition_name | -LDEV logical_device_# | -HELP]

LIST_CONTIGUOUS_BLOCKS returns information about the contiguous space available on a specified partition.

partition_name is the name of the partition. The maximum length of **partition_name** is 6 characters. If you do not have the partition name, you may obtain it from the STATUS DISKS display.

logical_device_# is the logical device number, which must be expressed in decimal format. Often, the logical device number is expressed in octal format. Use the TYPE command and the PRIMOS function [OCTAL] to convert octal to decimal. For example,

```
OK, TYPE [OCTAL 70]
56
OK,
```

When no argument is supplied, LCB returns information about your current attach point.

LCB is most useful on a partition containing CAM files because LCB determines the degree to which the partition is fragmented. CAM files require contiguous blocks of space.

LIST_CONTIGUOUS_BLOCKS Output

LCB returns information about the

- Sizes of the 20 largest blocks of free space larger than 32 records
- Total number of fragments (or free contiguous blocks)
- Minimum and maximum CAM file extension sizes in effect for that partition
- Total number of records
- Total number of free records or available space
- Percent full (free records divided by total records)

The following is an example of LCB output for the partition *INTEG*.

```
OK, LIST_CONTIGUOUS_BLOCKS INTEG
```

```
LIST_CONTIGUOUS_BLOCKS 22.0 INTEG

Largest 20 blocks of free                176      85
contiguous space larger                  175      84
than 32 records.                         152      83
                                         133      80
                                         108      72
                                         104      70
                                          99      62
                                          97      51
                                          91      50
                                          87      49

Number of fragments  : 3958      Total records     : 125920
Minimum extent size  :   16      Free records      :  16497
Maximum extent size  :   32      Percentage full   : 86.8
```

For more information, see *Operator's Guide to File System Maintenance*.

▶ LIST_DISKS [disk_names] [options]

Operator Command

LIST_DISKS displays information for either local or remote partitions that have been added to the system. It can also display the free record size of the local partitions and the IDs of current users. LIST_DISKS is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_DUMP [–HELP]

Operator Command

LIST_DUMP displays the current settings for a partial tape dump. These settings are the default parameters plus any additional parameters that have been set by either the DUMP_SEGMENT or DUMP_USER commands, or both. For more information, see the *Operator's Guide to System Commands*.

▶ LIST_EPF [pathname-1 [...pathname-8]] [options]

LIST_EPF displays information about a user's Executable Program Formats (EPFs).

EPF Types and Statuses

Each EPF has a type and a status. The three EPF types are the following:

- Program EPFs
- Program-class library EPFs
- Process-class library EPFs

The status of an EPF is one of the following:

- Mapped Active. These are EPFs that are currently mapped into your address space and are one of the following: program EPFs or program-class library EPFs that have been suspended while executing; or process-class library EPFs that have been initialized. (Initialized process-class library EPFs are also called in-use EPFs.)
- Mapped Nonactive. These are EPFs that are mapped into your address space but are neither suspended EPFs nor in-use EPFs.
- Not Mapped. These are EPFs that are not mapped into your address space but are stored in a directory.

Mapped EPFs (active or nonactive) are also referred to as open EPFs.

The LIST_EPF command works on any EPF, regardless of its type or status.

Specifying Pathnames

pathname is the pathname or filename of an EPF. You can specify a maximum of eight pathnames. You do not have to specify the EPF's suffix, which is either .RUN or .RP**n** (where **n** is a single digit). You can use wildcard names.

Unless you specify the –NOT_MAPPED option, LIST_EPF assumes that **pathname** is a mapped (active or nonactive) EPF. If **pathname** is not currently mapped into your address space or does not exist, the following message is displayed on your screen:

```
No entries selected.
```

If you issue LIST_EPF with no pathnames and no options, information is displayed on all the EPFs currently mapped into your address space.

LIST_EPF Output

The LIST_EPF default output displays the type and pathname of each EPF. The pathnames of the EPFs within each type are listed alphabetically.

The output pauses after each 23 lines of text and displays a `--More--` prompt. To display the next screen of output, press RETURN or enter Y, YES, OK, or NEXT. To exit from the command and return to PRIMOS command level, enter N, NO, Q, or QUIT. Use the –NO_WAIT option to suppress this prompt and enable continuous scrolling.

LIST_EPF Options

Use the following options to vary the selection and output display of LIST_EPF. You can specify more than one option if they do not conflict.

Option	*Meaning*
–ACTIVE	Displays information only on active EPFs.
–COMMAND_PROCESSING	Displays the pathnames of the EPFs and the state of the command processing features for program EPFs. These features include the type of file system objects on which the EPF may operate; whether the command processor should process wildcarding, treewalking, or command iteration; and the name generation position for the EPF.
–DETAIL	Displays all attributes for each entry selected.
–EPF_DATA	Displays the type, status, and pathname of an EPF, plus the following BIND information: the version of BIND used to create the EPF; the date on which the EPF was bound; the program name; the user version; the contents of the EPF comment field; and the number of debugger segments. If the EPF was bound by a version of BIND unable to indicate the first five BIND items, the command returns the message `EPF data not available.`
–HELP	Displays the syntax of the LIST_EPF command. The HELP screen is also displayed if PRIMOS detects an error while parsing the command.
–LIBRARY	Displays information only on program-class library and process-class library EPFs.
–NOT_ACTIVE	Displays information only on nonactive EPFs (that is, EPFs currently mapped into your address space, but not suspended program EPFs or in-use process-class library EPFs).
–NOT_MAPPED	Displays information only on unmapped EPFs (that is, EPFs that exist in your file system but are not currently mapped into your address space). If you do not specify **pathname**, information is displayed on all the unmapped EPFs in your current directory.
–NO_WAIT	Suppresses the `--More--` prompt and does not pause after every 23 lines of output, but scrolls the output continuously.

-PROGRAM

Displays information only on program EPFs.

-SEGMENTS

Displays the following information on all EPFs currently mapped into your address space: type, status, pathname, the absolute segment number used for EPF procedure, and the absolute segment/offset number of the linkage areas for the most recent invocation of the EPF.

To the left of the absolute segment number used for EPF procedure is a zero or a positive, even integer that relates the segment number to the relative segment number indicated by the same integer in the BIND map for the EPF. Correspondingly, to the left of the absolute segment/offset number of the linkage area used for the most recent invocation of the EPF is a negative, even integer that relates the segment number to the relative segment number indicated by the same integer in the BIND map for the EPF.

If the EPF has not yet been allocated either procedure segments or linkage areas, the message **not allocated** is displayed.

LIST_EPF Examples

The following examples illustrate the LIST_EPF command:

```
OK, LIST_EPF

1 Process-Class Library EPF.

(active)      <GRP3>LIBRARIES*>SYSTEM_LIBRARY.RUN

1 Program-Class Library EPF.

(not active) <GRP3>LIBRARIES*>FORTRAN_IO_LIBRARY.RUN

1 Program EPF.

(not active) <GRP3>CMDNC0>LD.RUN

OK, LIST_EPF -NOT_ACTIVE

1 Program-Class Library EPF.

(not active) <GRP3>LIBRARIES*>FORTRAN_10_LIBRARY.RUN

1 Program EPF.

(not active) <GRP3>CMDNC0>LD.RUN
```

```
OK, LE CIRCLE.RUN -DETAIL

1 Program EPF.

(not active) <GRP3>BEECH>CIRCLE.RUN
  1 procedure segment:     +0:4775
  1 linkage area:          -2:4777(3)/3074
  bind version:       21.0
  date of binding:    87-03-10.10:26:20.Tue
  program name:       CIRCLE
  user version:       (none)
  comment:            (none)
  debug segments:     1
  command options:    wldcrd,trwlk,iter     file,dir,segdir,acat     1

OK,
```

For further information on LIST_EPF and on EPFs, see the *Programmer's Guide to BIND and EPFs*.

See also REMOVE_EPF.

▶ LIST_GROUP

LIST_GROUP lists the ACL groups to which a user belongs.

An ACL group is a number of users who have the access rights assigned to that group. All group names begin with a period (for example, .MANAGERS). ACL groups are created by the System Administrator or by a Project Administrator. Each user may be a member of as many as 32 groups.

If you are not a member of any group, LIST_GROUP displays the following message:

```
OK, LIST_GROUP
No groups.
OK,
```

If you are a member of one or more groups, the command lists the group names, as in the following example:

```
OK, LIST_GROUP
Groups are:   .TEAM   .ADMINISTRATORS   .LEADERS
OK,
```

For further information on ACLs and groups, see the *PRIMOS User's Guide*.

▶ LIST_LAN_NODES [–lan-names] [{–HOST / –LTS}]

Operator Command

LIST_LAN_NODES displays information about the configured LAN300 (the Prime Local Area Network). LIST_LAN_NODES is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_LHC_STATUS [options]

Operator Command

LIST_LHC_STATUS displays status information pertaining to a LAN Host Controller 300 (LHC), a component of the Prime LAN300 network installation. LIST_LHC_STATUS determines the integrity of an LHC located either locally on the host system or remotely on the network.

For more information about the LAN300, see the *User's Guide to Prime Network Services*.

▶ LIST_LIBRARY_ENTRIES [pathname-1 [...pathname-8]] [options]

LIST_LIBRARY_ENTRIES displays the entrypoints in a library EPF.

The LIST_LIBRARY_ENTRIES command works on both program-class library EPFs and process-class library EPFs.

Specifying Pathnames

pathname is the pathname or filename of a library EPF. You can specify a maximum of eight pathnames. You do not have to specify the EPF's suffix, which is either .RUN or .RP**n** (where **n** is a single digit). You can use wildcard names.

If no pathnames are specified, information is displayed on all the library EPFs listed in your entrypoint search list (ENTRY$.SR).

LIST_LIBRARY_ENTRIES Output

The default LIST_LIBRARY_ENTRIES output for each EPF displays a two-line header followed by an alphabetical listing of the entrypoints. The first line of the header lists the EPF's status (active, not active, or not mapped) and its pathname. The

second line of the header lists the EPF's type (process-class or program-class), the total number of entrypoints, and the number of entrypoints currently selected for display.

The EPFs are listed alphabetically if there are two or more.

After each 23 lines of text, a `--More--` prompt is displayed. To display the next screen of output, press RETURN or enter Y, YES, OK, or NEXT. To exit from the command and return to PRIMOS command level, enter N, NO, Q, or QUIT. Use the –NO_WAIT option to suppress this prompt and enable continuous scrolling.

LIST_LIBRARY_ENTRIES Options

The options for LIST_LIBRARY_ENTRIES are as follows:

Option	*Meaning*
–ENTRYNAME entryname [...entryname-7]	Displays the entrypoint specified by **entryname**, which can be a wildcard name. You can specify a maximum of eight entrynames.
–HELP	Displays the syntax of the command. The Help screen is also displayed if PRIMOS detects an error while parsing the command.
–NO_WAIT	Suppresses the `--More--` prompt and does not pause after every 23 lines of output, but scrolls the output continuously.

The example below illustrates the LIST_LIBRARY_ENTRIES command:

```
OK, LLENT <GRP3>LIBRARIES*>FORTRAN_IO_LIBRARY -ENTRYNAME R@@

(active) <GRP>LIBRARIES*>FORTRAN_IO_LIBRARY.RUN
Program-Class Library EPF,  110 Total Entrypoints, 6 Selected Entrypoints

RDASC      RDBIN      RDLN$P     RDLN$P$I   RECLN$     RSTBL

OK,
```

For further information on LIST_LIBRARY_ENTRIES and on EPFs, see the *Programmer's Guide to BIND and EPFs*.

See also LIST_EPF; LIST_SEARCH_RULES; REMOVE_EPF; SET_SEARCH_RULES.

▶ LIST_LIMITS

LIST_LIMITS displays information on the various attributes of a user's command environment.

The System Administrator sets the command environment attributes of all users. The attributes are set in a user profile or on a systemwide basis. The limits of these attributes are important when you are programming with Executable Program Formats (EPFs). The limits include the following:

- The number of command levels you can use
- The number of programs you can invoke at any command level
- The number of private dynamic segments you can use
- The number of private static segments you can use

The following example illustrates the LIST_LIMITS display:

```
OK, LIST_LIMITS

Maximum number of command levels: 10
Maximum number of program invocations: 10
Maximum number of private static segments: 100
Maximum number of private dynamic segments: 150

OK,
```

For further information on LIST_LIMITS and on EPFs, see the *Programmer's Guide to BIND and EPFs*.

See also INITIALIZE_COMMAND_ENVIRONMENT.

▶ LIST_LTS_STATUS [options]

Operator Command

LIST_LTS_STATUS extracts and examines status information pertaining to a LAN Terminal Server (LTS), a component of the Prime Local Area Network (LAN) network installation. LIST_LTS_STATUS determines the integrity of an LTS located either locally on the host system or remotely on the network.

For more information about the LAN network installation, see the *User's Guide to Prime Network Services*.

▶ LIST_MEMORY [options]

Operator Command

LIST_MEMORY displays memory usage by the number of segments, resident pages, and wired pages per user process. Either user IDs or user numbers can be used to identify the user. LIST_MEMORY is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_MINI_COMMANDS [command_match]

LIST_MINI_COMMANDS lists the PRIMOS commands allowed at mini-command level.

The LIST_MINI_COMMANDS output gives the names and abbreviations of the PRIMOS commands you can use when you reach mini-command level, that is, when you exceed your maximum command level depth.

command_match is a character string used as a pattern match for the commands allowed at mini-command level. The character string can contain wildcard characters, such as @@.

If you do not specify **command_match**, all the commands allowed at mini-command level are listed alphabetically, as follows:

```
OK, LIST_MINI_COMMANDS

Abbrev  Full name                    Abbrev  Full name
- - -   - - - - - - - - - - - - - -   - - -   - - - - - - - - - - - - - - - -
C       CLOSE                        COMO    COMOUTPUT
DMSTK   DUMP_STACK                   ICE     INITIALIZE_COMMAND_ENVIRONMENT
LE      LIST_EPF                     LL      LIST_LIMITS
LMC     LIST_MINI_COMMANDS           LS      LIST_SEGMENT
        LOGIN                        LO      LOGOUT
P       PM                           PR      PRERR
        RDY                          REN     REENTER
RLS     RELEASE_LEVEL                REMEPF  REMOVE_EPF
S       START

OK,
```

For further information on LIST_MINI_COMMANDS and on EPFs, see the *Programmer's Guide to BIND and EPFs*.

▶ LIST_PRIMENET_LINKS [options]

Operator Command

LIST_PRIMENET_LINKS displays the status of PRIMENET links. A link can be a PRIMENET-configured SMLC line, a PRIMENET-configured node on a ring, or a LAN300 network. For each link, the command displays the node or Public Data

Network (PDN) to which it is connected, the number of active virtual circuits on the link, and the availability of the link for traffic routing. LIST_PRIMENET_LINKS is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_PRIMENET_NODES [options]

Operator Command

LIST_PRIMENET_NODES displays all PRIMENET-configured remote nodes, the paths to those nodes, and the permitted access to those paths. Several paths can exist on the same node. LIST_PRIMENET_NODES is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_PRIMENET_PORTS [options]

Operator Command

LIST_PRIMENET_PORTS displays the port assignments for the system. LIST_PRIMENET_PORTS is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_PRIORITY_ACCESS diskname

LIST_PRIORITY_ACCESS lists the priority Access Control List (ACL) of the specified disk.

The LIST_PRIORITY_ACCESS command is provided because a priority ACL can prevent you from accessing directories and from using the LIST_ACCESS command. To display the names of the partitions connected to your system, use the STATUS DISKS command.

The following example illustrates the use of LIST_PRIORITY_ACCESS on the partition named GALAXY:

```
OK, LIST_PRIORITY_ACCESS GALAXY
Priority ACL on partition "<GALAXY>":
    MERCURY:   ALL
    $REST:     NONE
```

▶ LIST_PROCESS [options]

Operator Command

LIST_PROCESS displays the characteristics of a specified user process. These characteristics include the user's process identity, attach points, abbreviation file, active command input and command output files, connect times, CPU and I/O times and limits, membership of project groups, and all active remote identities. LIST_PROCESS is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_QUOTA [pathname] [–BRIEF]

LIST_QUOTA lists current disk quota and storage information for a directory.

A **quota** is the maximum number of disk records that a directory can hold. The System Administrator usually sets quotas on top-level directories. Users who have access rights can set quotas on directories with the SET_QUOTA command. If a directory has a quota, it cannot obtain records that would cause it to exceed the quota. This restriction is enforced on the entire subtree. If various levels of a subtree have multiple quotas, then the most restrictive quota is enforced.

pathname specifies the directory for which you want quota information. If you do not specify **pathname**, information for your current directory is listed. To use LIST_QUOTA, you must have List (L) access to the target directory and Use (U) access to all higher-level directories.

The LIST_QUOTA default output displays information for both quota and nonquota directories. (A nonquota directory has a quota setting of zero.) The information consists of the following three items, each displayed on a separate line:

- The maximum number of records allocated for use by the directory (the quota) or a message stating that the directory is not a quota directory.
- The total number of records currently used by the entire subtree, including the target directory. This number must be less than the quota.
- The number of records currently used by the target directory.

The –BRIEF option summarizes the same information on a single line. The three items are indicated by the following labels: Max (for the maximum number of records allowed for the directory); Used (for the total number of records currently used by the directory); and Records (for the number of records currently used by the target directory). The name of the directory is displayed as the last entry on the line only if you specified its pathname with the command.

Examples of Using LIST_QUOTA

The following three examples illustrate the LIST_QUOTA command.

Example 1: This example shows quota information for the subdirectory STATS in the directory REPORTS:

```
OK, LIST_QUOTA REPORTS>STATS

Maximum records allowed on "REPORTS>STATS" = 500
Total records used = 425.
Records used in this directory = 28.
OK,
```

The output shows that the quota for the directory STATS is 500 records. The directory tree is currently using 425 records, 28 of which are used by the entries in STATS itself.

Example 2: This example uses the same directory as Example 1, but uses the –BRIEF option to produce the same information in an abbreviated form:

```
OK, LIST_QUOTA REPORTS>STATS -BRIEF
MAX:       500, Used:       425, Records:          28, REPORTS>STATS
```

Example 3: This example is similar to Example 1, but assumes that STATS does not have a quota:

```
OK, LIST_QUOTA REPORTS>STATS

"REPORTS>STATS" is not a quota directory.
Total records used = 425.
Records used in this directory = 28.
```

Information on the records used by both the subtree and by the target directory are listed as for a quota directory.

For more information on directory quotas, see the *PRIMOS User's Guide*.

▶ LIST_RBF [{–DETAIL / –SIZE}]

LIST_RBF lists the attributes of a ROAM master or slave file.

The LIST_RBF output includes the activation status of the file, the status of the recovery attributes, the date and time last saved, the usage of the file, and the ROAM file identifier.

The –SIZE option also displays the size, in halfwords, of the requested segment directory and of the complete ROAM file. The –DETAIL option is reserved for future use and has no effect at present.

For further details on LIST_RBF, see the *ROAM Administrator's Guide*.

▶ LIST_REMOTE_ID [–ON nodename]

LIST_REMOTE_ID displays the user's remote IDs.

Your remote IDs are established with ADD_REMOTE_ID commands and are kept on a remote-ID list. The list holds a maximum of 16 remote IDs, each for use on a different remote system. LIST_REMOTE_ID displays the contents of this list. The display includes all remote IDs, project IDs (if any), and the systems on which the remote IDs are used. Passwords are never listed.

The –ON option limits the output to remote IDs on the system specified by **nodename**.

The following example illustrates the LIST_REMOTE_ID command with and without the –ON option. The three remote IDs were set up with the ADD_REMOTE_ID command.

```
OK, LIST_REMOTE_ID
System   User id                          Project id
- - -    - - - -                          - - - - - -
SYSB     LAURA.B
SYSC     LAURA.C                          OAKTREE
SYSF     LAURA.F                          GRAND
OK, LRID -ON SYSC
System   User id                          Project id
- - -    - - - -                          - - - - - -
SYSC     LAURA.C                          OAKTREE
OK,
```

See also ADD_REMOTE_ID; REMOVE_REMOTE_ID.

▶ LIST_SEARCH_RULES [{ [listname1. . .listname16] }]
–NO_WAIT
–HELP

LIST_SEARCH_RULES lists the contents of search list(s) for a user's process. LIST_SEARCH_RULES allows the use of wildcards.

LIST_SEARCH_RULES lists the five system-defined search rule files by default. These five search lists are

- ATTACH$
- COMMAND$
- ENTRY$
- INCLUDE$
- BINARY$

When you either log in or otherwise initialize a process, PRIMOS establishes the special-purpose search lists noted above. (See the SET_SEARCH_RULES command, descibed later in this chapter, for information about modifying the contents of these search lists and creating additional search lists.) The LIST_SEARCH_RULES command allows you to display the contents of your active search lists, and shows the pathname of the search rules file used to create each search list.

The **listname** argument allows you to specify which search lists you wish to see. If you specify one or more list names, LIST_SEARCH_RULES displays those search lists in the order specified. You can specify a maximum of 15 list names. If you do not specify **listname**, LIST_SEARCH_RULES displays the five system-defined search lists mentioned above, plus all of the search lists that you have defined. The most recently set search list is displayed first.

The –NO_WAIT option causes terminal output to scroll continuously.

The –HELP option displays the command syntax and options. A HELP screen is also displayed if PRIMOS detects an error while parsing the command.

The following example illustrates the LIST_SEARCH_RULES command:

```
OK, LSR

List: ATTACH$
Pathname of template: <UPLAB>SEARCH_RULES*>ATTACH$.SR

    -added_disks

List: INCLUDE$
Pathname of template: <UPLAB>SEARCH_RULES*>INCLUDE$.SR

    [HOME_DIR]

List: BINARY$
Pathname of template: <UPLAB>SEARCH_RULES*>BINARY$.SR

    [HOME_DIR]

List: COMMAND$
Pathname of template: <UPLAB>SEARCH_RULES*>COMMAND$.SR

    cmdnc0
```

```
List: ENTRY$
Pathname of template: <UPLAB>SEARCH_RULES*>ENTRY$.SR

    -primos_direct_entries
    libraries*>system_library.run
    libraries*>fortran_io_library.run
    libraries*>application_library.run
    -static_mode_libraries
OK,
```

For more information on the Search Rules facility, see the *Advanced Programmer's Guide, Volume II: File System.* For further information on system and administrator default rules, see the *System Administrator's Guide, Volume III: System Access and Security.*

See also LIST_LIBRARY_ENTRIES; SET_SEARCH_RULES.

▶ LIST_SEGMENT [segno-1 [...segno-8]] [options]

LIST_SEGMENT displays information about the DTAR2 private segments that a user is currently using.

segno specifies the segment number (in octal) about which you want information. You can specify a maximum of eight segment numbers, separating each number with a space or comma. You cannot use wildcarding and iteration. If you do not specify any segment numbers, LIST_SEGMENT displays information on each segment currently in use in the static and dynamic segment ranges.

LIST_SEGMENT Output

If you specify no options, the command lists the segment numbers and your access rights to each segment. Segment numbers are displayed in ascending numerical order. The access rights and their codes are as follows:

Code	*Access Allowed*
NULL	No access allowed
GATE	Gate access
R	Read access
RW	Read and write access
RX	Read and execute access
RWX	Read, write, and execute access

The LIST_SEGMENT output pauses after each 23 lines of text and displays a `--More--` prompt. To display the next screen of output, either press RETURN or enter Y, YES, OK, or NEXT. To exit from the command and return to PRIMOS

command level, enter N, NO, Q, or QUIT. Use the –NO_WAIT option to suppress this prompt and enable continuous scrolling.

LIST_SEGMENT Options

The LIST_SEGMENT options are listed below. You can specify more than one option if they do not conflict.

Option	*Meaning*
–BRIEF	Displays only the total number of segments currently in use in each segment range.
–DYNAMIC	Displays information only on private dynamic segments.
–HELP	Displays the command's correct syntax. The HELP screen is also displayed if PRIMOS detects an error while parsing the command.
–NAME	Displays the name of any EPF associated with the segment. This option is valid only for private dynamic segments. An EPF is associated with a segment if the procedure or linkage areas for that EPF are assigned to that segment. If more than one EPF is associated with a given segment (as may be the case if several EPFs have their linkage areas allocated within the same segment), the EPF pathnames are arranged alphabetically on separate lines. If a given EPF uses more than one segment, the EPF pathname appears alongside the segment number and access rights for each segment used. If a dynamic segment is not associated with an EPF, **none** appears instead.
–NO_WAIT	Suppresses the **--More--** prompt and does not pause after every 23 lines of output, but scrolls the output continuously.
–STATIC	Displays information only on private static segments.

The following example illustrates LIST_SEGMENT without options:

```
OK, LIST_SEGMENT

1 Private static segment.
segment access
- - - - - - - -
4000    RWX

5 Private dynamic segments.
segment access
- - - - - - - -
4152    RX
4153    RX
4154    RX
4155    RX
4377    RWX

OK,
```

For further information on LIST_SEGMENT and on EPFs, see the *Programmer's Guide to BIND and EPFs*.

See also LIST_EPF; LIST_LIMITS.

▶ LIST_SEMAPHORES [options]

Operator Command

LIST_SEMAPHORES displays the value of all semaphores in use on the system. LIST_SEMAPHORES is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_SERVER_NAMES [-HELP]

LIST_SERVER_NAMES lists server names on the local system together with the user ID and member process numbers for each server. The LIST_SERVER_NAMES command also displays the total number of servers on the local node.

PRIMOS assigns a server name to every process (user processes as well as system processes) when you initialize the process. By default, PRIMOS places each terminal process or phantom process in its own server. A PRIMIX child process, however, is a member of its parent's server.

At PRIMOS Revision 22.0, these server names are used by InterServer Communications (ISC). ISC is a PRIMOS facility that provides for message exchange between two concurrently running servers.

This command lists server name information as follows:

```
OK, LIST_SERVER_NAMES

[LIST_SERVER_NAMES Rev 22.0 Copyright (c) 1987, Prime Computer, Inc.]
5 servers for node SYS.01

Server Name          UserID        Member Process(es)

BBBBBBCVBBBC         SYSTEM          1
BBBBBBCWBBBG         EVELYN          22
RZBTDGZJBBDC         GLENN           29
RZBTDGZJBBGX         JOHN            31  32   33  34  40  41
                                     42  43
RZBTDGZJBBXZ         NANCY           30

OK,
```

Servers are listed in alphabetical order by server name.

For further information on server names, see the *Subroutines Reference V: Event Synchronization*.

See also LIST_SESSIONS.

▶ LIST_SESSIONS [–SERVER_NAME name [–SESSION number] | –DETAIL | –HELP]

LIST_SESSIONS lists Interserver Communications (ISC) sessions.

A session links two active servers. LIST_SESSIONS lists overview or detail information about a single ISC session, about all of the sessions belonging to a single ISC server, or about all ISC sessions. LIST_SESSIONS can list information only about sessions on the local node.

If you specify no options, LIST_SESSIONS displays overview information about all currently active ISC sessions. If there are no active sessions, LIST_SESSIONS returns the message `No Sessions`.

LIST_SESSIONS Options

The following LIST_SESSIONS options are available:

Option	*Meaning*
–SERVER_NAME name	Lists session information for all sessions owned by **name**. **name** is the name of an active server on the local node. **name** cannot be the ISC Network Server.
–SESSION number	Lists session information for session **number** owned by server **name**. You cannot use the –SESSION option without the –SERVER_NAME option.
–DETAIL	Lists detailed information for the sessions selected by the other LIST_SESSIONS options.
–HELP	Displays a summary of the LIST_SESSIONS options.

LIST_SESSIONS Output

LIST_SESSIONS displays two different levels of information about each session: overview information and detailed information. You can use either type of display for information about a single session, or for the sequential display of information about many sessions. Sessions are sorted by server name, then by session number within each server name. If you display information for all servers, each local session is represented by two entries, one for each server. A remote session is represented by only one entry for the server that is on the local system.

Overview Output

By default, the LIST_SESSIONS command displays overview information about each listed session. Session overview information is displayed as follows:

```
OK, LIST_SESSIONS

[LIST_SESSIONS Rev 22.0 Copyright (c) 1988, Prime Computer, Inc.]
NODE SYS.01 SESSIONS:

Server Name     Ses#   Init?   Phase    User ID
  Mate User ID                           Mate Node      Mate Server Name
-------------------------------------------------------------------------------
BBBBBBCWBBBG      1    yes     DXFER    GLENN
  NANCY                                    local node     RZBTDGZJBBDC

BBBBBBCXBBCD      3    no      ESTG     NANCY
  JOHN                                     SYS.04

RZBTDGZJBBDC      2    no      DXFER    NANCY
  GLENN                                    local node
```

This LIST_SESSIONS overview information contains the following parameters:

Parameter	*Meaning*
Server Name	The 12-character name of the server that owns this session.
Ses#	The session number that this server uses to identify this session. The server on the other side of the session identifies the session with a different session number.
Init?	If **yes**, the server is the session initiator. If **no**, the server is the session recipient.
Phase	The current state of the session. The following phase codes are possible: **DXFER** Data transfer **ESTG** Establishing **EXCPT** Exception pending **PEND** Pending **TERM** Terminating **UNDEF** Undefined
User ID	The user ID of this server.
Mate User ID	The user ID of the server on the other end of the session. If this user ID cannot be determined, this parameter displays a question mark **(?)** within parentheses.
Mate Node	The node name of the server on the other side of the session. If both servers are on the same node, this parameter displays **local node**. If this node name cannot be determined, this parameter displays a question mark **(?)** within parentheses.

Mate Server Name	The server name of the server on the other end of the session. This server name is only displayed if **Init?** = YES; otherwise, this field is blank.

Detail Output

If you specify the –DETAIL option, the LIST_SESSIONS command displays detailed information about each listed session. You can list detailed information about all sessions on the current node, or use the –SERVER_NAME or –SERVER_NAME and –SESSION options to select specific sessions for detailed display. Session detail information is displayed as follows:

```
OK, LIST_SESSIONS -DETAIL

[LIST_SESSIONS Rev 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Session #   1 of server BBBBBBCVBBBC

   Session detailed state:                    ESTABLISHED
   Mate user ID:                                    NANCY
   Mate node name:                           local node

Normal service:      CONFIGURED    Expedited service:  NOT
CONFIGURED
Max ctl:  128   Max data:  2048    Max expedited size:       0

Normal Send          Normal Recv          Message area
-----------          -----------          ------------
Queue size   7       Queue size   7       Block size        512
Free slots   7       Msgs on Q    0       Number of blocks  100
Total sent   0       Total recd   0       Current % in use:   0
                                          Max % in use:       0

Exp Send             Exp Recv
--------             --------
Queue size   0       Queue size   0
Free slots   0       Msgs on Q    0
Total sent   0       Total recd   0
```

This LIST_SESSIONS –DETAIL information contains the following parameters:

Parameter	*Meaning*
Session #	The number by which this server identifies the session. The server on the other side of the session identifies the session with a different session number.
Server	The 12-character name of the server that owns this session.
Session detailed state	The current state of the session. The possible states are as follows: **ESTABLISHED** **EXCEPTION** **EXCEPTION BEFORE ACCEPT RECEIVED**

```
PENDING
REQUEST RECEIVED
SESSION ACCEPTED
SESSION REJECTED
SESSION REQUESTED
TERMINATED
TERMINATED BEFORE REPLYING
TERMINATED PENDING
TERMINATING
UNDEFINED
```

Mate user ID
The user ID of the server on the other end of the session. If this user ID cannot be determined, this parameter displays a question mark **(?)** within parentheses.

Mate node name
The node name of the server on the other side of the session. If both servers are on the same node, this parameter displays **local node**. If this node name cannot be determined, this parameter displays a question mark **(?)** within parentheses.

Normal service
CONFIGURED indicates that normal message service is provided for this session; **NOT CONFIGURED** indicates that normal message service is not provided for this session.

Expedited service
CONFIGURED indicates that expedited message service is provided for this session; **NOT CONFIGURED** indicates that expedited message service is not provided for this session.

Max ctl
The maximum length (in bytes) of the control part of normal messages.

Max data
The maximum length (in bytes) of the data part of normal messages.

Max expedited size
The maximum length (in bytes) of expedited messages.

Queue size
The queue length for normal and expedited message send and receive queues.

Free slots
The number of free message slots available on each of the send queues.

Total sent
The number of normal and expedited messages successfully sent.

Msgs on Q
The number of messages on each receive queue.

Total recd
The number of normal and expedited messages successfully received.

Block size
The length in bytes of the blocks that comprise the session's data part message area.

Number of blocks
The number of blocks that comprise the session's data part message area.

Current % in use	The percentage of the session's data part message area that is currently in use.
Max % in use	The maximum percentage of the session's data part message area that has been used since the beginning of this session.

To display online help information on LIST_SESSIONS, use the following command:

```
OK, LIST_SESSIONS -HELP
```

For further details on ISC sessions, refer to the *Subroutines Reference V: Event Synchronization.*

See also LIST_SERVER_NAMES.

▶ LIST_SYNC [line numbers] [options]

Operator Command

LIST_SYNC displays the configuration of all enabled synchronous lines. Be aware that LIST_SYNC does not return the status of these lines: the software subsystems controlling the line (such as PRIMENET, RJE, and DPX) perform this function. LIST_SYNC is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide.*

▶ LIST_TAPE [pathname] –MT n [options]

LIST_TAPE lists information about the contents of tapes created by the ARCHIVE, BACKUP, or TRANSPORT commands.

Before using LIST_TAPE, you must assign a tape drive unit with the ASSIGN command and mount the correct reel on the drive. The number of the drive unit is specified by **n** in the –MT **n** keyword.

By default, the LIST_TAPE display includes the types and pathnames of files, directories, segment directories, and access categories recorded on the tapes. More details can be listed with LIST_TAPE options.

Only system operators or the System Administrator can list BACKUP tapes. Only the tape owner or the System Administrator can list ARCHIVE tapes. Any user can list TRANSPORT tapes.

To display online help information on LIST_TAPE, use the following command:

```
OK, LIST_TAPE -HELP
```

For details on LIST_TAPE and its options, see the *Data Backup and Recovery Guide.*

▶ LIST_UNITS [options]

Operator Command

LIST_UNITS either displays the open file units for any system user, the current attach points of all users on the system, or the user IDs of those users with either a particular file unit open or a file unit open in a particular directory. LIST_UNITS is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LIST_VAR [variable-names]

LIST_VAR lists variables and their values from an active global variable file.

If you specify no variable names, LIST_VAR lists all global variables and their values, as in the following example:

```
OK, list_var
  .AWAY                          BEECH>BRANCH2>TWIG4
  .HOME                          BEECH>BRANCH5>TWIG3
  .ADAM                          GARDEN>ADAM
OK,
```

If you specify one or more variable names, only those variables and their values are listed. Separate the variable names with a space or a comma, as in the following example:

```
OK, LIST_VAR .AWAY .HOME
  .AWAY                          BEECH>BRANCH2>TWIG4
  .HOME                          BEECH>BRANCH5>TWIG3
OK,
```

You can specify a wildcard name as a variable name, as follows:

```
OK, list_var .a@@
  .AWAY                          BEECH>BRANCH2>TWIG4
  .ADAM                          GARDEN>ADAM
OK,
```

For further information on global variables, see the *PRIMOS User's Guide*.

See also DEFINE_GVAR; DELETE_VAR; SET_VAR.

▶ LIST_VCS [options]

Operator Command

LIST_VCS displays the state of virtual circuits. If you use LIST_VCS without any options, all active virtual circuits on the system are displayed. LIST_VCS is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ LL

See LIST_LIMITS.

▶ LLENT

See LIST_LIBRARY_ENTRIES.

▶ LMC

See LIST_MINI_COMMANDS.

▶ LOAD

LOAD invokes LOAD, the Prime linking loader for R-mode binary files.

The LOAD utility converts R-mode binary (object) files into R-mode executable runfiles. The R-mode binary files are produced by the FTN, PMA, and RPG compilers.

To load V-mode or I-mode segmented code, use the BIND or the SEG command. For a complete discussion of the loaders and an example of the use of LOAD, refer to the *LOAD and SEG Reference Guide*.

▶ LOGIN [user-id [password]] [–ON nodename] [–PROJECT project-id]

LOGIN admits a user into a system.

Use LOGIN to begin a work session at a terminal. After you log in, you are connected to a directory set by the System Administrator as your Initial Attach Point. This directory is your origin directory.

user-id is a name with a maximum of 32 characters created by the System Administrator. The first character of the ID must be a letter, and the rest may be any combination of letters, digits, periods (.), underscores (_), and dollar signs ($). Case does not matter because PRIMOS converts lowercase letters to uppercase. Your user ID identifies you to your local system and to other systems that recognize IDs defined on your system. You can specify only an existing user ID. The user ID you specify is not required to be the name of your origin directory.

password, if required for your system, is a string of 16 or fewer ASCII characters. The password cannot contain PRIMOS reserved characters. (See the CHANGE_PASSWORD command for a list of the PRIMOS reserved characters.)

How you specify your password depends on how your system is configured. On some systems, you can include the password on the command line. Other systems prompt you for the password after you issue the LOGIN command, in which case your terminal does not echo the password.

–ON nodename specifies the remote system on which you wish to log in. **nodename** is usually the PRIMENET node name of that system.

–PROJECT project-id specifies the project ID with which you are associated after login. Your System Administrator determines if you are required to give a project ID at login. Project IDs conform to the same naming rules as user IDs.

If you issue the LOGIN command without specifying any arguments, PRIMOS prompts you for your user ID and, depending on the configuration of your system, for your password and project ID. If your System Administrator has defined a default project for you, you are not prompted for a project ID.

Changing Passwords

At Revision 22.0 or later, PRIMOS may require you to change your password periodically. Your System Administrator sets your password lifetime; that is, the frequency with which your password must be changed.

Also, at Revision 22.0 or later, the System Administrator can require all passwords to be computer generated. In this case, you receive a computer-generated password when your password expires or system passwords change. Only you and the System Administrator know your password. See your System Administrator for more details.

Failed Login

If your attempt at login is not successful, PRIMOS often displays a message explaining why you could not log in. If your attempts repeatedly fail, make a note of the message and see your System Administrator.

When you log in, you are notified of the number of failed login attempts under your user ID since your last successful login. For example, if there had been three unsuccessful login attempts under a users ID since the last successful login, the following message would appear after a successful login:

```
Warning!  There were 3 failed attempts to login under this ID since the
          last successful login.
```

After PRIMOS displays a message like the one above, it resets to 0 the count of failed login attempts since your last successful login.

Note

If you are sure that the failed login attempts were not yours, contact your System Administrator.

If there were no failed login attempts, PRIMOS does not give you a message when you login.

Remote Login

Remote logins are accomplished with the –ON option. The remote system must be linked to your system by PRIMENET, and you must have the following information:

- The name of an existing ID on the remote system
- The PRIMENET node name of the remote system
- Any required passwords or validation codes

You are logged in to your local system if you either omit the –ON option and give a user ID that is valid on the local system, or use the –ON option with the node name of your local system and give a user ID that is valid on the local system. If you specify an incorrect node name, the PRIMENET connection is broken.

Login Over Login

You can log in to a system as another user without having to first log out your present user ID. Using this login-over-login method, you can either log in as a different user on the present (local) system, or you can log in remotely to another system. (In order to use login-over-login, the configuration directive LOGLOG must be enabled on your system.)

If you wish to log in as a different user on your current system, issue the LOGIN command with the desired user ID as you would normally. In this case, PRIMOS logs out your original user ID and then prompts you to supply the password of the new user ID. Having done so, you are now logged in as a different user with the same user number. If you subsequently wish to log in again on the present (local) system employing your original user ID, you may do so without having to log out first.

If you wish to log in as a user on a remote system, issue the LOGIN command with the desired user ID and –ON **sysname** option as you would normally. PRIMENET first logs you out of the local system, then connects you to the remote system. The remote system asks you for your password. You may be asked for a remote password as an additional security measure. (See your System Administrator for more details.) If you subsequently wish to log in to yet another system, or if you wish to log back in to the local system, you must log out of the remote system first.

LOGIN Examples

The following four examples illustrate the LOGIN command.

Example 1: This example illustrates a login procedure on a local system that requires a project ID:

```
LOGIN
User-id? CARROLL
Password? JABBERWOCK
Project-id? LOOKING_GLASS

CARROLL (user 27) logged in Wednesday, 17 Jun 87 10:47:04
Welcome to PRIMOS version 22.0
Copyright (c) Prime Computer, Inc. 1985.
Last login Monday, 15 Jun 87 11:49:32

OK,
```

Example 2: This example assumes a system that allows you to specify all login information on the command line, without your waiting for system prompts. You must specify your user ID before you can specify your password and/or other options.

```
LOGIN JULIA REDWING -PROJECT SAIL

JULIA (user 16) logged in Thursday, 18 Jun 87 09:06:16
Welcome to PRIMOS version 22.0
Copyright (c) Prime Computer, Inc. 1985.
Last login Thursday, 18 Jun 87 08:05:32

OK,
```

In the above example, user *JULIA* gives her password *REDWING* and the project ID *SAIL*. She will be a member of the project *SAIL* during her work session.

Example 3: This example illustrates a remote login by user *HARRY* onto a system with the node name *HQA*:

```
LOGIN HARRY -ON HQA
PRIMENET 19.4 HQA
Password? TUDOR

HARRY (user 7) logged in Friday, 19 Jun 87 08:32:12
Welcome to PRIMOS version 22.0
Copyright (c) Prime Computer, Inc. 1985.
Last login Monday, 15 Jun 87 10:50:48

Enter validation code: ARAGON

OK,
```

In the above example, user *HARRY* must supply a login password and a validation code (neither of which is echoed at the terminal when entered) before he can work on the remote system. These requirements vary depending on the configuration of the remote system.

Example 4: This example illustrates password lifetime. User *DIANA*'s password has expired. *DIANA* is given a computer-generated password.

```
LOGIN DIANA
Password? <password typed but not echoed>


Computer generated passwords are in effect.
Please ensure that you can view your new password in privacy.
Type RETURN to continue:  Return

Your new password is 'MUFEBEBA'.
Reenter new password for confirmation: <password typed but not echoed>

Your new password is confirmed.


DIANA (user 6) logged in Thursday, 14 Jan 88 13:59:16.
Welcome to PRIMOS version 22.0
Copyright (c) 1987, Prime Computer, Inc.
Serial #*ENA Master (US Software Engineering)
Last login Thursday, 14 Jan 88 11:48:24.


OK,
```

▶ LOGOUT [–usernumber]

LOGOUT ends a user's work session on the system.

LOGOUT is the last command you issue when giving up access to a system. During the LOGOUT procedure, PRIMOS performs the following functions:

- Closes all user files.
- Releases all devices assigned to your terminal, including Network Terminal Service (NTS) lines.
- Detaches you from your origin directory.
- Returns to the supervisor all segments that you were using.
- Displays a logout message at both your terminal and, optionally, at the supervisor terminal. The logout message includes the date and time of the logout and the total hours and minutes of your connect time, CPU time, and I/O time. For more details about time accounting, see the TIME command.
- If the line is an NTS line, performs a disconnect sequence.

Logging Yourself Out

To log yourself out, issue the LOGOUT command without an argument, as in the following example:

```
OK, LO

MARLOWE (user 35) logged out Wednesday, 29 Apr 87 11:47:52.
Time used: 00h 01m connect, 00m 02s CPU, 00m 01s I/O.

OK,
```

Logging Another User Out

If another user (such as a phantom or a process on another terminal) has the same login name as yours, you can specify a user number to log out that user. (Use the STATUS USERS commands to display the names and numbers of all users.) When you specify a user number, do not leave a space between the hyphen and the user number, as in the following example:

```
OK, lo -68
```

When issued from the supervisor terminal, the command logs out the user specified by the user number, regardless of the user's login name.

Forced Logout

You may be logged out by the system if there has been no activity at your terminal for longer than the maximum inactive time allowed. When this force logout occurs, the normal logout message is preceded by the following message:

```
***From PRIMOS: maximum inactive time limit exceeded.
```

If you are logged out by a command issued at the supervisor terminal, the message `forced logout` is displayed, followed by the normal logout message.

▶ LON [{ -ON / -OFF }]

LON controls phantom logout notification.

When you initiate a phantom process, the phantom is logged in under your user ID. When the phantom is finished, PRIMOS automatically logs it out and notifies you. The following is an example of a phantom logout message:

```
Phantom 99: Normal logout at 10:33
Time used: 00h 02m connect, 00m 07s CPU, 00m 10s I/O
```

The LON command allows you to defer these messages or to display them immediately.

Deferring Phantom Logout Messages

To defer the reception of phantom logout notification messages, specify the –OFF argument, as follows:

```
OK, LON -OFF
```

Instead of displaying the messages at your terminal immediately, PRIMOS stores the messages until you issue a LON –ON command.

You may want to defer logout messages when you are using a text editor or running a COMOUTPUT file.

Receiving Phantom Logout Messages

To display logout notification messages that have been deferred by a LON –OFF command, use the –ON argument, as follows:

```
OK, LON -ON
```

The accumulated messages are displayed all at once.

The command also sets your terminal to receive subsequent phantom logout messages immediately.

The –ON argument is the default when you log in. Logout notification is sent to your terminal unless you specifically prevent it by using the –OFF argument.

Example of LON

The following example illustrates the use of the –OFF and –ON options. After issuing the LON –OFF command, you initiate three phantoms. The user numbers that are assigned to the phantoms are displayed, as follows:

```
PHANTOM is user 110
PHANTOM is user 115
PHANTOM is user 107
```

When these phantoms are logged out, the logout notification messages are stored by PRIMOS. To display these messages, issue the LON –ON command, as follows:

```
OK, lon -on

Phantom 110: Normal logout at 11:40
Time used: 00h 00m connect, 00m 00s CPU, 00m 00s I/O.

Phantom 107: Abnormal logout at 11:40
Time used: 00h 00m connect, 00m 03s CPU, 00m 01s I/O.

Phantom 115: Normal logout at 11:40
Time used: 00h 00m connect, 00m 04s CPU, 00m 01s I/O.
```

Phantoms and phantom logout are discussed in more detail in the *PRIMOS User's Guide*.

▶ LOOK [–usernumber [segnumber [access [mapseg]]]]

Operator Command

LOOK provides access to any segment in the system. For details on LOOK, see the *System Administrator's Guide, Volume I: System Configuration* and the *Operator's Guide to System Commands*.

▶ LOOPBACK [options]

Operator Command

LOOPBACK sends data packets between target components on the network and reports the results to the network user and to the Distributed System Management (DSM) facility. This loopback operation is used to verify the integrity of the network software operation and the network path between the target components.

LOOPBACK is used with the Prime LAN300 (Local Area Network) facility. For more information, see the *User's Guide to Prime Network Services*.

▶ LPAC

See LIST_PRIORITY_ACCESS.

▶ LQ

See LIST_QUOTA.

▶ LRID

See LIST_REMOTE_ID.

▶ LS

See LIST_SEGMENT.

▶ LSR

See LIST_SEARCH_RULES.

▶ M

See MESSAGE.

▶ MAX

See MAXUSR.

▶ MAGNET [options]

MAGNET reads, writes, and copies magnetic tapes that are not in Prime format to and from Prime disk files.

Using MAGNET, you can read files from disk or tape, write files to disk or tape, and create output spool files. The MAGNET options are

Option	*Meaning*
–OPERATOR	Displays mount or dismount messages at the supervisor terminal. Do not use with the –USER option.
–OVERWRITE	Overwrites a BRMS-labeled tape. Without this option, the only BRMS tapes MAGNET can use are those labeled as free tapes.
–SILENT	Displays only severity 2 and 3 error messages.
–USER	Displays mount or dismount messages at the user's terminal. Do not use with the –OPERATOR option.

Note

MAGNET is not an archiving or system backup facility. For archiving, use ARCHIVE/ARCHIVE_RESTORE or MAGSAV/MAGRST. For system backup, use BACKUP/BACKUP_RESTORE, MAGSAV/MAGRST, or PHYSAV/PHYRST.

For complete information on the operation of MAGNET, see the *MAGNET User's Guide*.

▶ MAGRST [options]

MAGRST restores into the PRIMOS system information from a magnetic tape created by MAGSAV.

When invoked, MAGRST responds with a series of questions. These questions and the appropriate user replies are discussed in the *Data Backup and Recovery Guide*, which also details the operation of MAGRST.

▶ MAGSAV [options]

MAGSAV writes a PRIMOS disk file, directory tree, or disk partition to 7-track or 9-track magnetic tape.

MAGSAV is a PRIMOS disk-to-tape backup subsystem. When invoked, MAGSAV responds with a series of questions. These questions and the appropriate user replies are fully explained in the *Data Backup and Recovery Guide*, which also gives complete information on the options and operation of MAGSAV.

See also ARCHIVE; BACKUP; MAGRST.

▶ MAKE –DISK pdev –PARTITION diskname [options]

Operator Command

MAKE formats disks for use by PRIMOS. For detailed information on the operation and options of MAKE, see the *Operator's Guide to File System Maintenance*.

▶ MAXSCH n

Operator Command

MAXSCH sets the variable MAXSCH in the system database named SUPCOM, which controls the circumstances in which the backstop process (part of the scheduler) adds processes to the ready list. **n** is the value of the variable MAXSCH. (The default value depends upon such factors as the amount of main memory and the type of CPU.) For details on MAXSCH, see the *Operator's Guide to System Commands*.

▶ MAXUSR [n]

Operator Command

MAXUSR controls the number of users that are allowed to be logged in to PRIMOS at any time. **n** is the maximum number of allowable users. (The default is the number of configured users.) For details on MAXUSR, see the *Operator's Guide to System Commands*.

▶ MCLUP

The MCLUP command is now obsolete and has been replaced by the MPLUSCLUP command.

▶ MDUMP

MDUMP invokes the MDUMP utility that dumps a MIDASPLUS file into a sequential disk file.

When invoked, the MDUMP program asks you a series of questions about the order, contents, and format of the file dump. For details on the operation of MDUMP, see the *MIDASPLUS User's Guide*.

▶ MEDUSA workstation-name

MEDUSA brings up a PRIME MEDUSA workstation.

PRIME MEDUSA, the Prime CAD/CAM system for design and drafting, is a separately priced product that requires special hardware and terminals. The MEDUSA command executes the command input files that initialize a workstation for use with the PRIME MEDUSA software.

▶ MEDCONFIG [project-name] [option]

MEDCONFIG configures a PRIME MEDUSA installation. **option** is either –ADMIN or –BASE. If you do not supply a project name, MEDCONFIG prompts you for one. For details on MEDCONFIG, see the *PRIME MEDUSA Guide to System Management*.

▶ MEDUTIL [project-name]

MEDUTIL runs PRIME MEDUSA utility programs. If you do not supply a project name, MEDUTIL prompts you for one.

MESSAGE [{user-id / –usernumber}] [options]

MESSAGE sends a one-line message to another user and controls the receive state of a user's terminal.

The MESSAGE command performs the following functions:

- Sends a message to the operator or to other users, either on your system or on remote systems
- Changes the receive state of your terminal
- Checks the receive state of your terminal or of another user's terminal
- Allows the operator to communicate with all or specific users, or with operators of other systems

The command format of MESSAGE depends on which of these functions you are performing.

Sending Messages

Use MESSAGE to communicate with another user (either local or remote) or with the operator.

Sending Messages to Other Users on Your System

To send a one-line message to another user on your system, use the following format:

MESSAGE {**user-id** / **–usernumber**} [–NOW]

user-id is the name under which the recipient is logged in. If two or more users are logged in under **user-id**, all of them receive the message.

–usernumber is the number of a specific terminal. You must supply the number with an initial hyphen (for example, –32). If you specify a user number, only the terminal with that number receives the message. (Use either the STATUS USERS or the MESSAGE –STATUS command to display a list of all user IDs and user numbers currently logged in.)

The –NOW option displays the message immediately on the recipient's screen. If you do not specify –NOW, the message is displayed when the recipient returns to PRIMOS command level.

After you issue the MESSAGE command, the cursor moves to the beginning of the next line so that you can enter your message. The text of the message must be no more than 80 characters in length and must be on one line.

When displayed on the recipient's terminal, the message is prefaced with your user ID and the time the message was sent. This format is

```
***user-id (user n on systemname) at hh:mm
  text-of-message
```

user-id is your login name, **n** is your user number, **systemname** is the name of your system, and **hh:mm** is the time of day in hours and minutes.

Example of Sending a Local Message: In the following example, user DONNA on system SYS.B sends a message to user HOPE on the same system:

```
OK, MESSAGE HOPE -NOW
Marketing meeting has been changed to 3:00 PM tomorrow.
OK,
```

The message is displayed at HOPE's terminal as follows:

```
*** DONNA (user 15 on SYS.B) at 13:54
Marketing meeting has been changed to 3:00 PM tomorrow.
```

Sending Messages to the Operator

To send a message to the supervisor terminal, use the following format:

MESSAGE [-1] [-NOW]

The operation of the -NOW option and of the rest of the MESSAGE procedure is the same as described above.

Sending Messages to Remote Users

If your local system is attached to a network, use the following format to send a message to a user on a remote system:

MESSAGE { user-id / -usernumber } -ON nodename [-NOW]

You must know the PRIMENET node name of the remote system to which you are sending the message, as well as the user ID or user number of the recipient.

The rest of the remote MESSAGE procedure is identical to the local procedure, as is the operation of the -NOW option. If used, -NOW must be the last word on the command line.

For example, to send a message to user QUEEN on a system with PRIMENET node name SYS.E, issue the following command:

```
OK, MESSAGE QUEEN -ON SYS.E -NOW
```

Then enter the message when the cursor moves to the beginning of the next line.

Note

If you send a message from the local supervisor terminal to a local user who is logged in remotely to another system, you must specify the –NOW option. If the –NOW option is not specified, the message is not sent to the intended recipient, and an error message is returned to the supervisor terminal.

Changing the Receive State of a Terminal

The MESSAGE command also controls the flow of messages by setting the receive state of a terminal. To change the receive state, use the command format

MESSAGE option

option is one of the following:

Option	*Meaning*
–ACCEPT	Allows reception of all messages. This is the default state when you log in.
–DEFER	Allows reception only of deferred messages. Messages sent with the –NOW option are rejected; other messages are displayed when you return to command level.
–REJECT	Rejects all messages.

Setting your terminal's receive state to defer or reject messages is necessary if you do not want messages to interrupt a terminal session. For example, you might change your receive state just before printing a file because a message received during printing would also be printed with the file's contents.

You cannot send deferred or immediate messages when your receive state has been changed by a MESSAGE –REJECT command. Nor can you send an immediate message (with the –NOW option) when your receive state has been changed by a MESSAGE –DEFER command. The reason for these restrictions is that a recipient could not respond to your message.

Checking the Receive State of a Terminal

Use the –STATUS option to check the receive state of a user's terminal (including your own). The –STATUS output lists the specified user's name, number, and receive state. Use the –ON option to determine the receive state of a user on a remote system.

The command format for checking the receive state of another terminal is

```
MESSAGE –STATUS [ { user-id      } ] [–ON nodename]
                { usernumber }
                { ME         }
```

The meanings of the options and arguments are

Option	*Meaning*
–STATUS	Lists the receive state of all users on your system.
–STATUS **user-id**	Lists the receive state of all users on your system with the login name of **user-id**.
–STATUS **usernumber**	Lists the receive state of the terminal with the number **usernumber**. Do not use an initial hyphen before the user number.
–STATUS ME	Lists the receive state of your terminal. You cannot use the –ON option with –STATUS ME.
–ON **nodename**	Lists the receive state of users on a remote system named **nodename**, when used with any of the –STATUS forms except –STATUS ME. The remote system must be linked to your own system through a network.

Examples of Checking Receive States

The following examples illustrate the use of the –STATUS option of the MESSAGE command.

Example 1: This example illustrates checking the status of all users:

```
OK, MESSAGE -STATUS

User                                    No  State
SYSTEM                                   1  Accept
LEWIS                                    2  Accept
DDM                                      3  Reject
DIXON                                    4  Accept
ABIE                                     5  Defer
ARTHUR                                   6  Accept
SANCHEZ                                  7  Accept

OK,
```

Example 2: This example illustrates checking the receive state of a specific user on your system:

```
OK, MESSAGE -STATUS ABIE

User                                    No  State
ABIE                                     5  Defer

OK,
```

Example 3: This example illustrates checking your own receive state:

```
OK, MESSAGE -STATUS ME

User                                    No  State
SANCHEZ                                  7  Accept

OK,
```

Example 4: This example illustrates checking the receive state of a user on a remote system:

```
OK, M -STATUS E_JONES -ON SYS.D

User                                    No  State
E_JONES                                 22  Defer

OK,
```

Using MESSAGE as an Operator Command

Operators can use MESSAGE to send messages to a specific user, to all users, or to an operator of a remote system on the network. The operator form of MESSAGE has additional arguments and options. For information on using MESSAGE as an operator command, see the *Operator's Guide to System Commands*.

Error Messages

Listed below are some error messages you may receive when using MESSAGE. Each error message is followed by an explanation.

Improper command usage or arguments. (MESSAG)
The command contains an invalid use of options, such as –ON and –STATUS ME.

Option "-option" not recognized by this command. (messag_)
The option specified by **-option** is invalid.

*****Unknown addressee.**
The command specifies the user ID or user number of a user who is not logged in.

*****User n not receiving now. (MESSAG)**
This message means one of two things: if you sent a message with the -NOW option, the recipient's receive state is either DEFER or REJECT; if you sent a message without the -NOW option, the recipient's receive state is REJECT.

*****Unknown node (PRIMENET). (SMSG$)**
You specified a system name unknown to your network.

*****User n busy, please wait. (SMSG$)**
User **n** already has a deferred message waiting. Only one deferred message is allowed.

*****Requires -ACCEPT enabled.**
You must issue a MESSAGE -ACCEPT command before sending messages.

*****Requires -ACCEPT or -DEFER enable.**
You must issue a MESSAGE -ACCEPT or MESSAGE -DEFER command before sending messages.

▶ MIRROR_OFF pdev_1 pdev_2 [options]

Operator Command

MIRROR_OFF stops the mirroring of the two mirrored partitions and shuts down either the primary partition **pdev_1**, or the secondary partition **pdev_2**, or both. For more information, see the *Operator's Guide to File System Maintenance*.

▶ MIRROR_ON pdev_1 pdev_2 [options]

Operator Command

MIRROR_ON starts the mirroring of two partitions, **pdev_1** and **pdev_2**. The two partitions to be mirrored must be identical in size and location on their respective physical disks. For more information, see the *Operator's Guide to File System Maintenance*.

▶ MONITOR_NET [options]

MONITOR_NET invokes the PRIMENET Information Monitoring (PIM) utility.

The PIM utility displays information on synchronous lines, virtual circuits, and the ring. The information includes performance, traffic, and status data. Use the following command to display the syntax and options of MONITOR_NET:

MONITOR_NET -HELP

For details on the operation and options of MONITOR_NET, see the *Operator's Guide to Prime Networks*.

▶ MPACK

MPACK invokes an interactive MIDASPLUS utility for packing and restructuring MIDASPLUS data and index subfiles.

The utility recovers space left by deleted data subfile entries, unlocks locked data records, and restructures the data subfile to correspond to the order of entries in the primary index. For details, see the *MIDASPLUS User's Guide*.

▶ MPLUSCLUP [{ –USER **usernumber** / –ALL }]

MPLUSCLUP performs cleanup for MIDASPLUS after an abnormal program termination.

The cleanup procedure closes any open MIDASPLUS files and releases the internal resources held by the MIDASPLUS user. The resources include buffers, internal MIDASPLUS locks, and record locks.

When invoked without an option, MPLUSCLUP cleans up only for the user who issued the command.

When invoked with the –USER option, MPLUSCLUP cleans up only for the user specified by **usernumber**. (Use the STATUS USERS command to list user numbers.) The –USER option releases the specified user's internal resources, but does not release record locks and does not close files. If **usernumber** is not your own, you must issue the command from the supervisor terminal.

The –ALL option can be issued only from the supervisor terminal. When used with this option, MPLUSCLUP releases all internal resources held by MIDASPLUS users. Record locks are not released and MIDASPLUS files are not closed. Program execution can usually continue from the point of interruption.

For further details on MPLUSCLUP, see the *MIDASPLUS User's Guide*.

▶ MRGF file-a file-b [file-c ...file-e] –OUTF out-file [options]

MRGF merges as many as five ASCII files into a single output file. **file-a** is the original file (that is, the common ancestor of **file-b** through **file-e**). MRGF assumes that changes have been made in **file-a** to produce the other files.

The –OUTF keyword and its **out-file** argument are mandatory and specify the pathname of the output file. All filenames, including **out-file**, may be specified by entryname or by pathname.

The MRGF options are as follows:

Option	*Meaning*
–BRIEF	Does not display the lines that differ between the files, only the file identification letters and the line numbers of the differing lines.
–FORCE	Does not ask the user to resolve conflicts between files, but automatically writes the changes in **file-b** to **out-file**. Edit mode is not entered and no manual selection is allowed by the user.
–MINL **number**	Sets the minimum number of lines that must match (after a difference in the files is found) in order to resynchronize all file merging. (The default is 3.)
–REPORT **pathname**	Writes to a file (named **pathname**) the lines of text that differ between files. Resolvable differences (that is, those resolved automatically by MRGF) are not displayed at the terminal. User-resolvable differences (that is, those requiring manual selection by the user) are displayed as well as being written to **pathname**.

MRGF Operation

If you invoke MRGF without options, it works as follows. The first file, **file-a**, is compared line by line with each of the other files (**file-b** through **file-e**). Lines that match in all files are copied into **out-file**.

When MRGF finds differences between files, it attempts to resynchronize the files. Rematching is complete only when a minimum number of lines match in all files. The default of this minimum number is 3, but you can change it with the –MINL option.

After resynchronization is complete, the selection of the lines to be written to the output file takes place. If only one file differed from **file-a**, the changes in that file are copied into **out-file**. If all files differed identically from **file-a**, those changes are also copied. This type of selection is called **automatic selection** because MRGF does not require verification by the user.

If, however, conflicting changes are found in several files (or if only one file is being merged with the original), MRGF prints the differing lines at the terminal. Each line is prefaced with an identification letter (letters A through E for **file-a** through **file-e**, respectively) and a line number.

MRGF then enters Edit mode to allow you to resolve the conflict. Select the lines you want written to **out-file** by entering one of the MRGF subcommands listed below. This type of selection is called **manual selection**. After selection (either automatic or manual) is completed, the line-by-line comparison continues.

Edit Mode

As explained above, when MRGF encounters differences among the files that it cannot resolve, it displays the differing lines and then enters Edit mode so that you can manually select the line that is to be written to **out-file**. There is no prompt for Edit mode. Instead, MRGF prints the word "EDIT" and leaves the cursor at the beginning of the next line.

Use one of the MRGF subcommands listed below to make your selection. Issue the subcommands either in uppercase or lowercase letters.

Subcommand	*Function*
A	Inserts all of the differing lines from **file-a** into **out-file**.
B	Inserts all of the differing lines from **file-b** into **out-file**.
C	Inserts all of the differing lines from **file-c** into **out-file**.
D	Inserts all of the differing lines from **file-d** into **out-file**.
E	Inserts all of the differing lines from **file-e** into [out-file].
A**n**	Inserts line **n** of **file-a** into **out-file**.
B**n**	Inserts line **n** of **file-b** into **out-file**. (Similarly for **file-c** through **file-e**; for example, C**n**).
A**m,n**	Inserts lines **m** through **n** of **file-a** into **out-file**. (Similarly for **file-b** through **file-e**; for example, B**m,n**.)
PA	Prints all of the differing lines from **file-a**. (Similarly for **file-b** through **file-e**; for example, PC.)
PA**n**	Prints line **n** of **file-a**. (Similarly for **file-b** through **file-e**; for example, PD**n**.)
PA**m,n**	Prints lines **m** through **n** of **file-a**. (Similarly for **file-b** through **file-e**; for example, PE**m,n**).
OOPS	Undoes all previous editing for this discrepancy.
blank line	Enters Input mode.
GO	Terminates editing and proceeds with the merge.
QUIT	Terminates editing, closes all files, and exits from MRGF.

Input Mode

During Edit mode, you can insert new text into the output file by entering Input mode. To enter Input mode, enter a blank line (that is, press the RETURN key without typing anything). When MRGF enters Input mode, it prints the word "INPUT" and leaves the cursor at the beginning of the next line.

All text entered in Input mode is inserted into **out-file**. To return to Edit mode from Input mode, enter a blank line.

You cannot perform text editing on text entered in Input mode. Tab character expansion is not performed on lines entered in Input mode.

Line Length

The MRGF command operates on compressed lines of any length and assumes that files of common ancestry contain lines compressed in identical fashion. It is possible, however, for a mismatch to occur between two lines that appear identical but were compressed differently.

MRGF Example

This example illustrates the operation of MRGF by merging the following three files:

```
FOX.1          FOX.2          FOX.3
The            The            The
quick          quick          quick
brown          red            brown
fox            fox            fox
jumps          jumps          jumps
over           over           over
the            the            the
lazy           sleeping       snoring
dog            dog            dog
```

FOX.1 is the original file (**file-a**). FOX.2 is **file-b** and FOX.3 is **file-c**. The three files will be merged to produce a fourth file named FOX.4. The merge process is illustrated below:

```
OK, MRGF FOX.1 FOX.2 FOX.3 -OUTF FOX.4
[MRGF 19.4]

A8        lazy
CHANGED TO
B8        sleeping
BUT ALSO CHANGED TO
C8        snoring
EDIT.
B
GO

MERGE FINISHED.
1 MANUAL CHANGE.
1 AUTOMATIC CHANGE AS FOLLOWS:
       1 FROM FILE B

OK,
```

The merged output file (FOX.4) contains the following text:

```
The
quick
red
fox
jumps
over
the
sleeping
dog
```

The one manual change occurred when the user issued the command B, which specified that line B8, rather than lines A8 or C8, was to be inserted into FOX.4. MRGF made one automatic change, which was to insert the word *red* from FOX.2 into FOX.4. MRGF made this choice because FOX.2 differed from FOX.1, whereas FOX.3 was identical.

If the –FORCE option were specified in this example, the same merged output would be produced. However, line B8 from FOX.2 would be inserted automatically because FOX.2, which is **file-b**, is the preferred file. The user would not be queried in this instance.

Purpose of MRGF

The MRGF command, like the CMPF command, is invaluable for parallel software development. The MRGF command allows automated merging of program changes, and eliminates the need for editing of programs when two (or more) sets of changes made to a program are to be combined. However, you should check the resulting merged output file carefully before you use it.

MRGF is especially useful for combining changes to a program that have been made in parallel by several programmers. It can also be useful for distributing software changes to one or more sites, or to one or more persons.

▶ MTRESUME MT**n pathname [options]**

Operator Command

MTRESUME is an internal command used by the operator to invoke (resume) certain .SAVE utility programs from magnetic tape in MAGSAV format. **n** specifies the logical mag tape unit number, and **pathname** identifies the file to resume.

For a complete description of MTRESUME and its options, see the *Operator's Guide to System Commands*.

▶ NET option

Operator Command

NET assigns, activates, and deactivates half-duplex (HDX) network connections. For details, see the *PRIMENET Planning and Configuration Guide*.

▶ NETLINK [options]

NETLINK connects the user to another networked system.

The NETLINK command invokes the NETLINK utility which allows you to make as many as six remote login connections to other networked systems. The other systems must be connected to your system through PRIMENET or through a non-Prime network, such as a Packet Switched Data Network (PSDN). If your system has a PSDN link, NETLINK can access any system in the network, including non-Prime systems.

When you issue the NETLINK command without options, the utility enters Command mode and displays an at sign (@) as a prompt. You must then issue a NETLINK command to continue. Among the tasks NETLINK commands allow you to accomplish are the following:

- Connect to and use a maximum of six different remote systems at the same time
- Transfer files across networks
- Set data transmission characteristics
- Display the status of your connection
- Specify the various fields of the connect packet when data transmission characteristics of a foreign system differ from that of a Prime system

The following section describes the basic procedure for using NETLINK. For detailed information on the operation, options, and error messages of NETLINK, see the *User's Guide to Prime Network Services*.

Using NETLINK

The basic procedure for using NETLINK is described below:

1. Enter NETLINK's Command mode by issuing the NETLINK command without options. When Command mode is entered, the @ prompt appears.
2. Connect to the remote system by issuing one of the following two command formats:

```
C address
NC address
```

address is either the host address assigned by the Packet Switched Data Network or a PRIMENET system name. Use C to connect to another system in a PRIMENET network or to reverse the charges over a PSDN, as in the following examples:

```
@ C SYS14

@ C 2080:12300011
```

Use NC when you do not want to reverse the charges. (NC is required for many international calls.)

When the connection is established, the following message is displayed:

```
address Connected
```

address is the address you specified with the C or the NC command.

3. Log in to the system as you would normally, entering any validation codes as required.
4. After you finish the terminal session, log out as you would normally. The message **address Disconnected** appears. (**address** is the address you specified in step 2.) NETLINK Command mode is reentered and the @ prompt is displayed.
5. You may now connect to another site or return to PRIMOS. To return to PRIMOS, enter the QUIT command. To connect to another site, repeat step 2 with the appropriate address.

NETLINK Example

In the following example of a basic terminal session, user NOVAK connects to the remote system named SYS.D, which requires the validation code CURVE:

```
OK, NETLINK
[NETLINK Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]

@ C SYS.D

SYS.D Connected
PRIMENET 21.0 SYS.D
LOGIN NOVAK
Password?  TANAGER

NOVAK (user 48) logged in Wednesday, 24 Jun 87 15:14.52.
Welcome to PRIMOS Version 21.0
Last Login Tuesday, 23 Jun 87 18:33.21.


Enter validation code:  CURVE
```

```
OK,
  .
  .          /* continue with normal terminal session
  .
OK, LOGOUT

NOVAK (user 48) logged out Wednesday, 24 Jun 87 17:34.44.
Time used: 02h 20m connect, 00m 12s CPU, 00m 02s I/O.

Wait...


SYS.D Disconnected

@ QUIT

OK,
```

▶ NSED [pathname]

NSED invokes the nonshared version of EDITOR, the Prime line-oriented text editor.

If you specify **pathname**, that file is loaded into the text buffer and the nonshared editor begins in Edit mode. If no pathname is specified, the editor starts in Input mode with an empty buffer.

Use NSED in VCP mode because it does not support the shared EDITOR. The nonshared editor is identical in operation to ED, the shared EDITOR. For detailed information on EDITOR, see the *New User's Guide to EDITOR and RUNOFF.*

▶ NTS_ASSOCIATE

Operator Command

$$\left\{\begin{array}{l} \text{–LINE n –LTS_NAME name –LTS_LINE n} \\ \qquad\qquad\qquad\qquad \text{[–PERMANENT]} \\ \text{–HELP} \end{array}\right\}$$

NTS_ASSOCIATE maps a physical line on an LTS300 controller to a PRIMOS line number. The PRIMOS line number is a decimal value.

NTS_ASSOCIATE is used with the Prime Network Terminal Service (NTS). For more information, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

▶ NTS_LINE {–COMMAND / –HELP}

NTS_LINE places a terminal that is connected to a LAN Terminal Server 300 (LTS300) into Command mode. This action temporarily suspends your PRIMOS session and allows you to communicate instead with the LTS300. To resume the connection with PRIMOS, give the LTS command CONTINUE. The NTS_LINE command is particularly useful when the LTS escape character has been disabled.

NTS_LINE is used with the Prime Network Terminal Service (NTS). For further information, see the *NTS User's Guide*.

▶ NTS_LIST_ASSOCIATE [{–LINE n / –LTS_NAME name –LTS_LINE n / –HELP}]

NTS_LIST_ASSOCIATE displays the current associations (if any) of assignable NTS PRIMOS line numbers with their corresponding LTS300 name or line numbers. You can use the command to display the association for a specific PRIMOS line (with the –LINE option), or all lines on all LTS300s (with no option), or a specific LTS300 (with the –LTS_NAME option), or a specific line on an LTS300 (with the –LTS_LINE option). –HELP displays online help for the user.

NTS_LIST_ASSOCIATE is used with the Prime Network Terminal Service (NTS). For more information, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers*.

▶ NTS_UNASSOCIATE

Operator Command

{–LINE n / –LTS_NAME name –LTS_LINE n / –HELP}

NTS_UNASSOCIATE breaks the mapping of an associated LTS300 line with a PRIMOS line number in the NTS assigned line range. An association cannot be broken if the line is currently assigned.

NTS_UNASSOCIATE is used with the Prime Network Terminal Service (NTS). For more information, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers*.

▶ NUMBER

NUMBER numbers or renumbers statements in a BASIC program.

For NUMBER to work correctly, the BASIC commands must be in uppercase characters. The maximum line length that NUMBER can handle is 75 characters, plus 5 for the line number. Lines longer than 75 characters are truncated. If a statement is not numbered, NUMBER numbers it in order of its occurrence in the program.

Using NUMBER

When NUMBER is invoked, it displays the following prompt with the cursor at the beginning of the next line:

```
INTREENAME, OUTTREENAME, START, INCR,
```

The meanings of the four parameters are

Parameter	*Meaning*
INTREENAME	The pathname of the input file. The input file is the file that contains the BASIC program with the statements to be renumbered.
OUTTREENAME	The pathname of the output file. If no output filename is specified, the output is written to the input file.
START	The number to be given to the first statement. The number can be from 1 to 9999, inclusive. If you omit START, the value 1 is assumed. You must specify a value for START if you are going to specify a value for INCR.
INCR	The number by which each subsequeunt statement is to be incremented. The incremental value can be from 1 to 9999, inclusive. If you omit INCR, the value 1 is assumed. To specify a value for INCR, you must specify a value for START.

Only the first parameter (INTREENAME) is mandatory. If you enter one or more subsequent parameters, you must enter them all on the same line, separating each parameter with a comma or a space. Filenames can be entered either in uppercase or lowercase.

Example of Using NUMBER

This example uses the following BASIC program, which is stored in a file named NUM.IN:

```
OK, slist num.in
11 PRINT 'Enter the first number: '
12 INPUT A
23 PRINT 'Enter the second number: '
36 INPUT B
40 LET C = A * B
57 PRINT 'The answer is ',C
58 IF C <> 0 THEN 11
99 END
OK,
```

The sequence of commands shown below renumbers the program. The first statement starts at 10, and the number of each subsequent statement is incremented by 5. The renumbered program is stored in a file named NUM.OUT:

```
OK, number
INTREENAME, OUTTREENAME, START, INCR,
num.in num.out 10 5
OK,
```

The program is renumbered as follows:

```
OK, slist num.out
  10 PRINT 'First Number: '
  15 INPUT A
  20 PRINT 'Second Number: '
  25 INPUT B
  30 LET C = A * B
  35 PRINT 'Answer is ',C
  40 IF C <> 0 THEN 10
  45 END
OK,
```

Error Messages

Some of the NUMBER error messages are shown below. Each error message is followed by an explanation.

DECODE FORMAT/DATA MISMATCH
A non-numerical value was specified for START or INCR.

n DUP LINE NUMBER
There is more than one statement with the line number **n**.

Illegal name. BAD PARAMETERS
An illegal filename was specified.

INPUT FILE NULL
The specified input file is empty.

LINE NUMBER OVERFLOW
A new line number is greater than 9999. This error occurred because the START and/or the INCR parameters were too large.

MEMORY OVERFLOW
There is not enough memory to contain a map of line numbers.

Not found. BAD PARAMETERS
The specified input file does not exist.

▶ OAS

OAS starts up the Prime Office Automation System (OAS).

The OAS command logs you in to the OAS Master Function Selection, which allows you access to all Word Processing and Management Communications and Support options.

For detailed information, see the following documents:

- *OAS Word Processing Guide (PT65)*
- *OAS Word Processing Guide (PST 100 and PT45)*
- *OAS Management Communications and Support Guide*
- *Using OAS on the PT25 Management Workstation*

▶ OA_ADMIN

Operator Command

OA_ADMIN logs the Office Automation System (OAS) Administrator into Prime's OAS Administrator Option Selection. The OAS Administrator can then create and maintain master OAS files. For detailed information, see the *OAS System Administrator's Guide*.

▶ OPEN [pathname] funit key

OPEN opens a file on a specified file unit.

funit is a number from 1_8 to 177_8, inclusive, that specifies the file unit on which the file is opened. (Under special circumstances, PRIMOS may allow you to open file units larger than 177_8.) To name the file, specify a pathname before **funit**.

key is an octal number that determines the status of the file when opened. The number is the logical OR of three octal numbers that specify the New File, Action,

and Reference keys. The New File key represents the type of file. The Action and Reference keys specify the action to be taken with the file. The valid values of these keys are as follows:

New File Key	*Type of File Created*
0000	Sequential file (SAM)
2000	Direct access file (DAM)
4000	SAM segment directory
6000	DAM segment directory
10000	Directory (Avoid this; use the CREATE command instead.)
Action Key	*Action Taken*
1	Open for reading.
2	Open for writing.
3	Open for reading and writing.
4	Close.
5	Delete. (Avoid this; use DELETE command instead.)
6	Test for existence.
7	Rewind.
10	Truncate at current position. (Do not use a pathname.)
Reference Key	*Action Taken*
0	Enter in current directory.
100	Enter in segment directory open on **funit**.
1000	Change open mode of **funit**.

Example of Using OPEN

The following example shows how to use OPEN to open a DAM file for writing in the current directory. The file is named PAGE and is opened on File Unit 6.

The first step is to obtain a value for **key**:

2000	File is a direct access file (New File key).
0002	File is opened for writing (Action key).
0000	File is in the current directory (Reference key).
2002	The resulting value for **key**.

The second step is to use the OPEN command to open the file PAGE on File Unit 6. The STATUS UNITS command shows the results:

```
OK, open page 6 2002
OK, status units
```

```
User CALVERT                                    SYS4

File   File       Open     File
Unit   Position   Mode     Type   RWlock   Treename
   6   000000000     W      DAM    NR-1W   <BOOK>CHAPTER>PAGE


OK,
```

Use the CLOSE command to close a file that was opened with the OPEN command.

See also CREATE; INPUT.

▶ OPRPRI {0 1}

Operator Command

OPRPRI (operator privilege) sets or removes the protection on the LOOK command.

At Rev. 21.0, you are no longer required to use the OPRPRI command with the SHARE command. For further information, see the *Operator's Guide to System Commands*.

▶ ORIGIN

ORIGIN returns you to your origin directory from your current directory.

Your **origin directory** (also called your Initial Attach Point) is the directory to which you are first attached when you log in. Your **current directory** is the directory to which you are attached at any given moment. (Your origin directory is also your current directory when you are attached to it.) When you use the ATTACH command to attach to another directory, that directory becomes your current directory. Use the ORIGIN command to return to your origin directory from your current directory. ORIGIN does nothing if issued from your origin directory.

ORIGIN takes no command line arguments or options. For more information on ORIGIN, see the *PRIMOS User's Guide*.

▶ OWLDSC [–FAST] [–NOLOCK] [–REPORT]

OWLDSC invokes the OWL interface program, which allows an OWL-1200 terminal to emulate an IBM 3277 Model 2 Display Station on systems where DPTX/DSC is running. For details, see the *Distributed Processing Terminal Executive Guide*.

▶ P

See PM.

▶ PASCAL pathname [options]

PASCAL loads the Prime PASCAL compiler and compiles an object program from an ASCII source file named **pathname**.

It is recommended that you give **pathname** a .PASCAL suffix. Use the BIND or the SEG command to create a runfile from the object file. (BIND is recommended.)

For detailed information on the operation and options of PASCAL, see the *Pascal Reference Guide*.

▶ PASSWD [owner-password [nonowner-password]]

PASSWD changes the owner and nonowner passwords of the current password-protected directory.

owner-password is the new owner password that replaces the previous one. If you do not specify **owner-password**, the new owner password is blank.

nonowner-password is the new nonowner password that replaces the previous one. You cannot specify this argument unless you also specify **owner-password**. If you do not specify **nonowner-password**, the new nonowner password is null.

Guidelines for Using PASSWD

The following are some guidelines to keep in mind when using the PASSWD command:

- To use PASSWD, you must be the owner of the directory whose passwords are to be changed and you must be attached to the directory. A nonowner cannot give this command.
- The PASSWD command applies only to password-protected directories. If you use PASSWD on a directory protected by an Access Control List (ACL), the password will be in effect only after you give the REVERT_PASSWORD command.
- Passwords may be of any length, but they are matched by the first six characters only.
- To specify a lowercase password, enclose the password in single quotation marks.
- Do not confuse PASSWD with the CHANGE_PASSWORD command, which changes login passwords.
- The MAGRST command cannot restore directories with passwords unknown to the MAGRST user.

PASSWD Example

The following is an example of using PASSWD. The directory, named STUART and owned by the user, has an owner password of SATURN and a nonowner password of PLUTO. The user attaches to the directory and changes both passwords:

```
OK, ATTACH STUART SATURN
OK, PASSWD LION TIGER
OK,
```

STUART now has LION as its owner password and TIGER as its nonowner password.

▶ PASSWORD_DIRS {-ON / -OFF}

Operator Command

PASSWORD_DIRS enables or disables the ability to create password directories. For more information, see the *System Administor's Guide, Volume III: System Access and Security*.

▶ PHANTOM pathname [{funit / CPL-arguments}]

PHANTOM runs a command input file or CPL program as a phantom process.

pathname is the CPL or command input file that is run by the phantom user. **pathname** may be a simple filename if the file is in your current directory. The file contains the sequence of commands and other necessary input data to complete a given job.

funit is an octal number that specifies the file unit on which a command input file is opened. (The default is File Unit 6.) Do not use **funit** for CPL programs. **CPL-arguments** passes CPL arguments to a CPL program. Both of these optional arguments are further explained below.

Running Command Input Files as Phantoms

Command input files run as phantoms must have LOGOUT as the last command in the file, so that the phantom user can log out properly. Using COMINPUT –TTY or COMINPUT –END as the last line may cause an abnormal termination of the phantom. If LOGOUT is not the last line, the phantom reports an abnormal ending when it finishes processing the file.

You can control the file unit on which the command input file is opened by specifying an octal value for **funit**. The default value for **funit** is 6.

Running CPL Programs as Phantoms

In a CPL program run as a phantom, the &RETURN directive at the end of the program is interpreted as a LOGOUT command. **CPL-arguments** specify arguments to be passed to the CPL program. These arguments are used by the program's &ARGS directive. **funit** cannot be specfied with CPL programs.

For further information on CPL programs, see the *CPL User's Guide*.

Phantom Operation

A **phantom process** is a job performed by a phantom user. A **phantom user** is similar to any other PRIMOS user, except that the phantom is not associated with a terminal. Therefore, during execution of a phantom process, all controlling input must be read from either a command input file or a CPL program instead of a user terminal. In addition, terminal output during a phantom process is suppressed unless a command output file has been opened with a COMOUTPUT command in the command input file or CPL program. Output is then written to the command output file.

Note

You must be attached to a directory on your local node in order to run a phantom process.

At any given moment, PRIMOS has only a fixed number of phantom processes available. The PHANTOM command searches for a free process into which the phantom user can log in. If no free processes are available, the following message is displayed at your terminal:

```
No phantoms are available.  FILENAME
```

If a process is available, the phantom user is logged in to your login directory. Then, the phantom feature of PRIMOS attaches to your current directory. If **pathname** is a command input file, it is opened on File Unit 6 (or on **funit**). If **pathname** is a CPL file, it is opened on an available unit. The phantom process then takes all further commands from the command file in accordance with COMINPUT or CPL operation.

The following example illustrates initiating a phantom process:

```
OK, PHANTOM STARS.CPL
PHANTOM is user 106
OK,
```

If a phantom user attempts to read input from a terminal, the process aborts and logs out the phantom user. The logout message and the following message are then displayed at the system terminal:

```
User 106:  Phantom requested terminal input.
```

Any error that aborts the phantom command file also causes the phantom user to log out.

When a phantom that you started logs out, it attempts to notify you that it has logged out. The following is an example of such a message:

```
Phantom 106: Normal logout at 10:33
Time used: 00h 02m connect, 00m 07s CPU, 00m 10s I/O
```

You receive this message only if you are still logged in on the terminal from which the phantom was started. In other cases, the messages can be recorded by a user program using the subroutines LON$R and LON$CN. (For further information on these subroutines, see the *Subroutines Reference III: Operating System.*)

Terminal output generated by the phantom user program or directed to the user terminal by system commands (such as LD) is ignored, unless the COMOUTPUT command is invoked in the phantom command file.

To monitor the status of phantom users, use the commands STATUS ME (for your phantoms) or STATUS USERS (for all phantoms on your system). These commands display the phantom's directory name and user number.

To stop a phantom user that was started at your terminal, use the following command format:

LOGOUT **–usernumber**

usernumber is the number of the phantom user as reported by the PHANTOM command. The number must be preceded by a hyphen (for example, –106).

If you log out and later log in to the same directory, you can use the STATUS and LOGOUT commands, as before, to control the phantom. Any phantom on your system can be logged out at the supervisor terminal.

A CPL program or command file may issue the PHANTOM command. Command files running in phantoms may also include PHANTOM commands (that is, you can chain phantom command files in a manner similar to command input command files).

See also JOB; LON.

▶ PHYRST [options]

Operator Command

PHYRST copies partitions to disk from a magnetic tape created by PHYSAV.

For a complete description of PHRYST and its options, see the *Data Backup and Recovery Guide.*

▶ PHYSAV [options]

Operator Command

PHYSAV copies to magnetic tape the contents of one or more assigned disk partitions.

For details on PHYSAV and its options, see the *Data Backup and Recovery Guide*.

▶ PL1 pathname [options]

PL1 invokes the Prime full PL/I compiler and compiles an object program from the ASCII source file named **pathname**.

It is recommended that you give **pathname** a .PL1 suffix. Use the BIND or the SEG command to create a runfile from the object file. (BIND is recommended.)

For a full description of the PL/I compiler and its options, see the *PL/I Reference Guide*.

▶ PL1G pathname [options]

PL1G invokes the Prime PL/I Subset G compiler and compiles an object program from the ASCII source file named **pathname**.

It is recommended that you give **pathname** a .PL1G suffix. Use the BIND or the SEG command to create a runfile from the object file. (BIND is recommended.)

For a full description of the PL/I Subset G compiler and its options, see the *PL/I Subset G Reference Guide*.

▶ PLOT { pathname –AT destination [options] | –CANCEL plt-id | –LIST }

PLOT plots a metafile or device-specific plot file.

For PLOT to work, your system must have the Prime Plotter Software installed. PLOT also lists the contents of the local plot queue and removes one or more plot files from the plot queue. For details on PLOT, see the *Prime Plotter Software Guide*.

▶ PLOTOP { environment-name option | –STAT }

Operator Command

PLOTOP creates, starts, stops, and modifies plot despoolers.

For PLOTOP to work, your system must have the Prime Plotter Software installed. For details on PLOTOP, see the *Prime Plotter Software Guide*.

▶ PM

PM (Post Mortem) displays the contents of the RVEC user register vector, applicable only during the use of a static-mode program (not an EPF or a CPL program).

The RVEC parameters are described in Appendix A. PRIMOS first displays labels for the RVEC parameters, and then displays their values in the same order on the next line.

The PM command also displays the procedure base register (PB), the stack base register (SB), the link base register (LB), and the temporary base register (XB). These 32-bit registers are displayed at the user terminal on a text line separate from the other registers. Each of the Prime 350-class registers is displayed as two 16-bit octal numbers separated by a ring number and a slash (/) character.

The following example illustrates the PM command:

```
OK, PM
SA,EA,P,A,B,X,K=
1000 4543 1120 0 0 0 34000

PB,SB,LB,XB:
4000(3)/1120  4003(0)/4  4000(0)/5064  6(0)/14366
OK,
```

This example of PM under PRIMOS shows a PB of 4000(3)/1120, which indicates: ring 3, segment 4000_8. The word number portion of PB (1120 in this example) is the same value as the P parameter in the first line of PM's output. This number specifies the location within the segment at which the next instruction executes upon a possible receipt of a START command.

Note

PM does not give an accurate picture of the machine state of a program if the program was halted by an on-unit that does not allow the static-mode overseer to update the PM data. This situation would occur if a user on-unit returned to command level by calling COMLV$. The DUMP_STACK command, however, always produces an accurate display of the state of the program.

▶ PMA pathname [options]

PMA loads the Prime Macro Assembler and starts assembly of a source file specified by **pathname**.

It is recommended that you give **pathname** a .PMA suffix. You can specify more than one option if the options do not conflict. For a complete discussion of the assembler, including register settings, see the *Assembly Language Programmer's Guide*. For a complete listing of the PMA instruction sets, see the *System Architecture Reference Guide*.

▶ PRATIO { percent1 [...percent8] / -DISPLAY }

Operator Command

PRATIO changes the percentage of paging activity on the paging partitions currently installed on the system. The sum of the percentages for all paging partitions on the system must equal 100. See the *Operator's Guide to System Commands* for more information on specifying partitions and for more information about PRATIO.

▶ PRERR

PRERR either displays the ERRVEC message set by the ERRSET subroutine and the first six octal locations in ERRVEC, or displays ERRVEC and the last error message.

The PRERR command is useful only for debugging obsolete programs. On encountering an error condition, PRIMOS sets up an internal vector called ERRVEC with several pieces of information. One of these pieces is an error message. Refer to the *Subroutines Reference III: Operating System* for a description of ERRVEC.

With the system subroutine ERRSET (see the *Subroutines Reference III: Operating System*), a user program or routine can set the content of the error message. In addition, the program or routine can print the message, depending on whether the alternate return is set to zero or nonzero. If the user routine was the last routine to set ERRVEC, PRERR prints the user-stored message.

The following example illustrates the PRERR command:

```
OK, PRERR
17 1 0 74 134640 120240
OK,
```

Note

PRERR does not display useful information following the execution of an Executable Program Format (EPF) program or following such conditions as access violation faults or illegal segment number faults. In these cases, the needed information is printed as part of the system diagnostic for the error condition. It can also be obtained by using the DUMP_STACK command.

▶ PRIMAN

PRIMAN generates reports of system activity from files that have been produced by the PRIMON command. By invoking PRIMAN, you open the user interface to the PRIMAN screens in which you can construct a report request, execute the request, save it, or read another request. PRIMAN is a separately priced product. For more information, see the *PRIMAN User's Guide*.

▶ PRIME/SNA

PRIME/SNA is not a PRIMOS command, but is a group of separately priced products that support the interconnection of Prime systems with IBM SNA (Systems Network Architecture) networks. The initial PRIME/SNA products are the following two subsystems that supply the services of IBM devices connected to an SNA network:

Server Subsystem	Provides the services of one or more SNA Physical Unit Type 2 (PU.T2) secondary SDLC stations.
Interactive Subsystem	Provides emulation of devices in the IBM 3270 Information Display System family (3274 Control Unit, 3278 Display Station, and 3287 and 3289 Printers), in conjunction with the Server Subsystem.

The following six PRIMOS commands control the PRIME/SNA products:

Command	*Description*
SNADSC	User command for invoking IBM 3278 emulation and accessing IBM applications.
SNA_3270	Operator command for controlling the various functions of the PRIME/SNA Interactive Subsystem.
SNA_3270_CONFIG	Administrator command for creating, editing, or listing an Interactive configuration.
SNA_PRINT	Operator command for invoking the Interactive Printer Emulation facility.
SNA_SERVER	Operator command for controlling the various functions of the PRIME/SNA Server Subsystem.
SNA_SERVER_CONFIG	Administrator command for creating, editing, or listing the Server configuration.

These commands are briefly described in this chapter and are also described in detail in the following documents:

- *PRIME/SNA Installation and Configuration Guide*
- *PRIME/SNA Interactive Terminal User's Guide*
- *PRIME/SNA Operator's Guide*

To display online information on these commands, use the following command format:

HELP command-name

▶ PRIMON

PRIMON monitors current system activity through screens of bar graphs, or stores this data in a file that can be analyzed with the PRIMAN tool. PRIMON is a separately priced product. For more information, see the *PRIMAN User's Guide*.

▶ PRIMOS [pathname]

Operator Command

PRIMOS is a PRIMOS II command used by the operator or System Administrator to bring up the PRIMOS operating system. The PRIMOS command is now obsolete. For further information, see the *Operator's Guide to System Commands*.

▶ PRINT_SECURITY_LOG –LOGFILE pathname [options]

Operator Command

PRINT_SECURITY_LOG prints an audit trail report from an audit file. The audit trail is a record of system events; the System Administrator selects the types of events that are printed. For more information, see the *System Administrators Guide, Volume III: System Access and Security*.

▶ PRMPC pathname

PRMPC prints a file on an MPC parallel interface printer configured to PRIMOS.

pathname is the name of the file to be printed. Before issuing the PRMPC command, you must assign the printer (PR0) with the ASSIGN command.

▶ PROP { –STATUS [–ALL] [–NO_WAIT] | environment-name { –DISPLAY | –SUSPEND | –RELEASE | –BACK n } }

PROP displays information on system printers. As an operator command, PROP controls the environment of the spooler phantoms.

A spooler **environment** is the set of specifications under which the printer operates. Users can issue the –STATUS and –DISPLAY options to display information about the printers.

The –STATUS Option

–STATUS lists the names of the system printers that have active despooler phantoms. The –STATUS option format is

PROP –STATUS [–ALL] [–NO_WAIT]

–ALL displays the status of all environments for which environment files exist.

–NO_WAIT inhibits display pagination (that is, the **--More--** prompt is not displayed).

The following example illustrates the –STATUS option with the –ALL argument.

```
OK, PROP -STATUS -ALL
[PROP Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
DOC                 Idle
PLAIN               Idle
B1_LQP              Idle
MJD                 Not Started
B1_LQP_MJD          Not Started
OK,
```

In the example, the despooler phantoms controlling printers DOC, PLAIN, and B1_LQP are running, but those controlling MJD and B1_LQP_MJD are not. Though a user could spool a file to all five printers, the file would not print on MJD and B1_LQP_MJD until the operator started up their phantoms with PROP –START.

The –DISPLAY Option

–DISPLAY displays information on a specific printer environment. Use –DISPLAY to list all attributes about your printer.

The –DISPLAY option format is

PROP **environment-name** –DISPLAY

environment-name specifies the printer about which you are requesting information.

To find out the names of the printers on your system, use the PROP –STATUS command described in the previous section.

The following example illustrates the –DISPLAY output for the DOC printer.

```
OK, PROP PRINTA -DISPLAY
[PROP Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
DOC                 Idle
/*Doc.env, spool*, mjd, 25/02/88
/*Environment parameters for narrow line printer
/*Copyright (c) 1988, Prime Computer, Inc., Natick, MA 01760
```

```
PLL PRL
DEVICE DATAPRODUCTS  -EVFU_OFF
FORMAT -W 84 -1 66
LOG -ON
MESSAGE This is SYSA's DOC printer
ATT SYSA
ATT PRINT_ROOM
ATT WHITE
ATT DOC -MAN
HEADER 2
WARNING -OFF
CONFIG QUEUES
PRIORITY_SIZE 20
MAX_SIZE 50
/*
/*End of Environment parameters
OK,
```

For a full description of Printer Environment directives, see the *Operator's Guide to the Spooler Subsystem.*

The –SUSPEND Option

–SUSPEND halts a currently printing job. After receiving a suspension notice, PROP stops your print request at its current point and goes on to the next request in the spool queue. You may suspend any job under your user ID.

The –SUSPEND option format is

PROP environment-name –SUSPEND

The –RELEASE Option

–RELEASE releases from suspension (restarts) a job that was halted with the –SUSPEND option. –RELEASE restarts the job from the point where the job was originally halted. You may restart any halted job under your user ID.

The –RELEASE option format is

PROP environment-name –RELEASE

The –BACK n Option

–BACK n causes the despooler phantom to back up **n** number of pages and reprint the data from that point forward.

The –BACK n option format is

PROP **environment-name –BACK n**

where **n** is the number of pages that you specify the printer to go back.

The default value of **n** is one page.

–BACK n is useful when a large file is partially printed and the printer is interrupted. The –BACK n option allows the user to restart the printing of the file.

PROP as an Operator Command

The operator uses PROP to invoke the spooler management utility that controls the environment for a system printer. The operator format of PROP is

PROP **environment-name [option]**

For more information about the despooler phantom and for a complete list of printer environment commands, see the *Operator's Guide to the Spooler Subsystem.*

For more information on the operation and options of PROP as an operator command, see the *Operator's Guide to System Commands.*

▶ PROTEC pathname key1 key2

PROTEC is an obsolete command that sets protection rights on files. The PROTECT command, which is documented below, replaces the PROTEC command. For information on PROTEC, see Appendix C of this guide.

▶ PROTECT pathname [owner-code [nonowner-code]] [–REPORT]

PROTECT sets access rights for file system objects in a password-protected directory.

pathname specifies the file, subdirectory, or segment directory to be protected. If you use a wildcard name with no file type selection options, the default selects all objects. To use PROTECT, you must have owner access to **pathname**.

owner-code and **nonowner-code** designate the type of access rights the owner and nonowners will have to the object. Specify the rights with the following codes:

Code	*Description*
NIL	No access of any kind (default)
R	Read access only
W	Write access only
D	Delete access only
RW	Read and Write access only

Code	*Description*
RD	Read and Delete access only
WD	Write and Delete access only
RWD	All rights (Read, Write, and Delete)

If you specify neither **owner-code** nor **nonowner-code**, both owner and nonowner receive NIL access rights. The owner must therefore modify the protection rights before the files can be accessed.

The –REPORT option reports the results of each successful setting of protection rights. The following example illustrates the –REPORT option:

```
OK, PROTECT MAST>SAIL RWD R -REPORT
"MAST>SAIL" protected
OK,
```

Using PROTECT in an ACL Directory

When you use PROTECT with an object in an ACL directory, the object receives the designated rights but also retains its ACL rights. The ACL rights take precedence and the PROTECT rights are ignored when you access the object. If, however, you subsequently convert the directory to a password directory (with the REVERT_PASSWORD command), the protection rights set by PROTECT come into effect.

To use PROTECT in an ACL directory, you must have Protect (P) access to the directory.

▶ PRSER pathname

PRSER prints a file on the serial interface printer configured to PRIMOS.

pathname is the file to be printed. Before using PRSER, you must assign the printer (CENPR or CE2PR) with the ASSIGN command.

▶ PRTDSC station-1 [...station-n]

PRTDSC invokes the printer emulation program on systems where DPTX/DSC is running. Printer output is spooled with form type equal to the first six characters of the station name. For details, see the *Distributed Processing Terminal Executive Guide*.

▶ PRVER pathname

PRVER prints a file on a printer/plotter configured to PRIMOS.

pathname is the file you want printed. Before using PRVER, you must assign the plotter with the command ASSIGN PLOT.

The following example assigns the plotter and then prints the file SALES.GRAPH:

```
OK, ASSIGN PLOT
OK, PRVER SALES.GRAPH
```

▶ PSD

PSD loads and starts the R-mode version of the Prime Symbolic Debugger, an interactive debugging program.

When invoked, the PSD program displays a dollar-sign ($) prompt and waits for you to enter a PSD command. To return to PRIMOS, enter QUIT. For details on PSD, see the *Assembly Language Programmer's Guide*.

See also DBG; HPSD; PSD20; VPSD; VPSD16.

▶ PT45DSC

PT45DSC invokes the PT45 interface program that allows a PT45 terminal to emulate an IBM 3277 Model 2 display station on a system that is running DPTX/DSC. For details, see the *Distributed Processing Terminal Executive Guide*.

▶ PTELE

PTELE accesses the Office Automation System (OAS) Telephone Inquiry function. For detailed information, see the *OAS Management Communications and Support Guide*.

▶ R

See RESUME.

▶ RDY [options]

RDY sets the system prompts.

After PRIMOS executes a command, it displays a prompt message that indicates whether or not the command was executed successfully. There are three types of prompt messages, each of which has a brief form and a long form. These types and their default messages at login are as follows:

- Ready prompt. (Given after commands that executed successfully.) **OK,** is the default brief form. The default long form is **OK** followed by the clock time, CPU time used since the last prompt, I/O time used since the last prompt, the level number if you are at a command level above 1 (that is, if a program was interrupted by a CONTROL-P or by an error-handling mechanism), and a plus sign (+) if the level is static mode. (This means that the last command executed was an external command.) For example,

```
OK 13:18:24  7.736  2.593  level 2+
```

- Error prompt. (Given after commands that ran into fatal errors.) **ER!** is the default brief form. The default long form is **ER!** followed by the same information as the Ready long form.
- Warning prompt. (Given after commands that ran into error conditions but still executed to completion.) The default brief and long forms are the same as the Ready prompts.

The RDY command, when followed by one or more options, allows you to switch to the long forms of these prompts (or to the brief forms) or to change one or more of the six default messages.

If issued without an option, RDY displays the current Ready long-form prompt, as in the following example:

```
OK, RDY
OK 15:12:42  7.674  2.343  level 2+
OK,
```

The RDY options are listed below. **text**, required for some options, is a character string of no more than 80 characters in length. If **text** contains special characters or blanks, the entire character string must be enclosed within single quotation marks.

Option	*Function*
–BRIEF	Switches all prompts to the brief forms. (Default at login.)
–ERROR_BRIEF text	Changes the brief form of the Error prompt to **text**.
–ERROR_LONG text	Changes the text portion of the long form of the Error prompt to **text**. Clock, CPU, and I/O times (plus the level number and the plus sign, if applicable) are still displayed after **text**.
–LONG	Switches all prompts to the long forms.
–OFF	Suppresses the display of all prompt messages.
–ON	Reenables the display of all prompt messages. Unless the –LONG or –BRIEF option is given with the –ON option, messages appear in the format last specified.
–READY_BRIEF text	Changes the brief form of the Ready prompt to **text**.
–READY_LONG text	Changes the text portion of the long form of the Ready prompt fo **text**. Clock, CPU, and I/O times (plus the level number and the plus sign, if applicable) are still displayed after **text**.

-WARNING_BRIEF text	Changes the long form of the Warning prompt to **text**.
-WARNING_LONG text	Changes the text portion of the long form of the Warning prompt to **text**. Clock, CPU, and I/O times (plus the level number and the plus sign, if applicable) are still displayed after **text**.

The following series of RDY commands illustrates the effects of some of the RDY options:

```
OK, RDY -LONG  /* set prompts to long forms
OK 08:41:57  9.251  2.672
ZAP  /* this is an illegal command
Not found. ZAP (std$cp)
ER 08:42:21  0.212  0.000
RDY -BRIEF  /* reset prompts to brief forms
OK  RDY -RB 'Yes, Dave?'  /* change ready brief prompt
Yes, Dave? RDY -EB 'Sorry, Dave!' /* change error brief prompt
Yes, Dave? POW  /* enter an illegal command
Not found. POW (std$cp)
Sorry, Dave! RDY -RB 'OK,' -EB 'ER!' /* reset prompts to defaults
OK,
```

▶ REENTER

REENTER reenters a subsystem following a QUIT or an error condition.

For REENTER to succeed, the subsystem being reentered must define an on-unit for the condition REENTER$ that can go to the appropriate point within the subsystem. If no on-unit exists, the REENTER command fails and you return to PRIMOS command level.

▶ RELEASE_LEVEL [option]

RELEASE_LEVEL discards unwanted stack history.

The stack history, which is automatically saved by PRIMOS, is a record of the calls and returns created by user commands. **option**, which determines how much of the stack is released, is one of the following:

Option	*Meaning*
-ALL	Releases the entire stack down to listener level 1.
-TO **n**	Releases stack levels down to level **n**. **n** must be a positive decimal integer less than the current level number.
-LEVELS **n**	Releases **n** levels so that the new stack level will be the current level minus **n**. **n** must be a positive decimal integer such that the current level minus **n** is less than or equal to 1. (Default of **n** is 1.)

If you do not specify an option, one of the following results:

- If your most recent command was an internal command, the current level of the stack is released.
- If your most recent command was an external command, the history of that command is released but the level number of the stack is not changed.

To determine your current stack level, issue the RDY command without options. If you are above level 1, the level number is displayed. If you are at level 1, no level number is displayed. In either case, the long form of the Ready prompt is also displayed.

If the stack grows large and unwieldy as a result of too many interruptions, PRIMOS warns you to release the stack with the message

```
Now at command level n.  To release use RLS. (listen_)
```

▶ REMOVE_EPF [pathname] [options]

REMOVE_EPF removes an Executable Program Format (EPF) from a user's address space.

The EPF to be removed must be mapped into your address space and must not be suspended. (Use the LIST_EPF command to list the pathnames and statuses of your mapped EPFs.) REMOVE_EPF does not delete the EPF file itself. (To delete a mapped EPF, use REMOVE_EPF to unmap the EPF, and then use the DELETE command to delete the file.)

pathname is the name of the EPF you want removed. You do not have to include the EPF suffixes .RUN or .RP**n** (where **n** is a single digit). Both wildcarding and iteration can be used, but not treewalking.

If the EPF indicated by **pathname** is not currently mapped into your address space or does not exist, REMOVE_EPF displays the message

```
No EPFs removed (REMOVE_EPF).
```

If you omit **pathname**, REMOVE_EPF asks you to verify the removal of each EPF currently mapped into your address space. Enter Y or YES to remove the specific EPF, or enter N or NO to leave it mapped.

The REMOVE_EPF options are listed below. You can specify more than one option if the options do not conflict.

Option	*Meaning*
–ACTIVE	Removes only active (in-use) process-class library EPFs. Does not remove active suspended program EPFs.
–FORCE	Removes only active program-class library EPFs. Does not remove active program EPFs. Do not use the –FORCE option if an active program-class library EPF is to be called within the same program invocation by another EPF that has already established linkage to the active EPF.

-HELP	Displays the command's correct syntax. The HELP screen is also displayed if PRIMOS detects an error while parsing the command.
-NOT_ACTIVE	Removes only non-active EPFs (that is, EPFs that are currently mapped to your address space, but are neither suspended program EPFs nor active process-class library EPFs).
-NO_QUERY	Does not ask the user to verify that the EPF is currently in use within the user's address space.
-NO_VERIFY	Disables verification checking if wildcards are used within **pathname**.
-QUERY	Asks the user to verify that the EPF is currently in use within the user's address space. (Default)
-VERIFY	Asks the user to verify the removal of an EPF. (The default is not to require verification unless wildcards are used within **pathname**.)

The following example illustrates the REMOVE_EPF command without options:

```
OK, remove_epf
Ok to remove EPF file <OSGRP0>LIBRARIES*>FORTRAN_IO_LIBRARY.RUN? no
Ok to remove EPF file <OSGRP0>LIBRARIES*>FORTRAN_LIBRARY.RUN? no
Ok to remove EPF file <OSGRP0>LIBRARIES*>SYSTEM_LIBRARY.RUN? no
Ok to remove EPF file <OSGRP0>CMDNC0>HELP.RUN? no

No EPFs removed (REMOVE_EPF).
OK,
```

For details on REMOVE_EPF and on EPFs, see the *Programmer's Guide to BIND and EPFs* and the *Advanced Programmer's Guide, Volume I: BIND and EPFs.*

▶ REMOVE_PRIORITY_ACCESS diskname

Operator Command

REMOVE_PRIORITY_ACCESS removes a priority ACL from the disk partition specified by **diskname**.

For detailed information on REMOVE_PRIORITY_ACCESS and on priority ACLs, see the *System Administrator's Guide, Volume III: System Access and Security* or the *Operator's Guide to System Commands.*

▶ REMOVE_REMOTE_ID -ON nodename

Your remote-ID list contains the remote IDs that you can use to access files on remote systems. (Use the ADD_REMOTE_ID command to add a remote ID to the list.) The list can contain a maximum of 16 remote IDs, one ID per system. If your list has reached the 16-ID limit, you cannot add more remote IDs unless you first remove at least one remote ID with the REMOVE_REMOTE_ID command. To list the remote IDs on this list, use the LIST_REMOTE_ID command.

nodename is the name of the remote system whose remote ID you want removed. If **nodename** is not on the remote-ID list, the message `Not found` is returned.

The following example illustrates the removal of a remote ID from the system named T39:

```
OK, list_remote_id
System  User id  Project id
- - -   - - - -  - - - - - -
T35     FRED_J
T54     FRED_REM POWER
T39     FRED2    WISHBONE
OK, remove_remote_id -on t39
OK, list_remote_id
System  User id  Project id
- - -   - - - -  - - - - - -
T35     FRED_J
T54     FRED_REM POWER
OK,
```

▶ REN

See REENTER.

▶ REPLY options

Operator Command

REPLY sends messages to a user in response to requests for magnetic tape assignments. For details on the operation and options of REPLY, see the *Operator's Guide to System Commands.*

▶ RESET_DUMP [–HELP]

Operator Command

RESET_DUMP resets the parameters for a partial tape dump to the default values. Segments 0_8 through 1777_8 and 6000_8 through 6003_8 for all users are written to tape during a tape dump by default. For more information, see the *Operator's Guide to System Commands.*

▶ RESTOR pathname

RESTOR restores the R-mode runfile **pathname** from disk to memory, using the RVEC parameters saved with the file. To restore a 64V segmented mode runfile, use the RESTOR subcommand of SEG instead of the RESTOR command.

▶ REST_RBF source-pathname [target-pathname] [options]

REST_RBF activates (restores) an inactive ROAM file. RESTORE_RBF is a synonym for REST_RBF.

When you activate an inactive ROAM file, the recovery table (RCVTAB) is given the location of the ROAM master file and all its slaves. The file can then be accessed by runtime processing.

source-pathname is the name of the file to be restored. If a currently active version of this file exists, REST_RBF asks you if it should deactivate that file. If, however, you used the NO_QUERY option, REST_RBF deactivates the file without asking you. **target-pathname** is the filename or pathname of the new location for the active file. If **target-pathname** is an existing file, it is overwritten only if it is a ROAM master segment directory. If you do not specify **target-pathname**, REST_RBF executes a restore in place.

REST_RBF has the following options:

Option	*Meaning*
–CAM	Restores all the subfiles of the target segment directory as contiguous files. Cannot be used with the –DAM option. If neither –CAM nor –DAM is specified, the subfiles of the restored segment directory assume the same file type as those of the source segment directory.
–DAM	Restores all the subfiles of the target segment directory as DAM files. Cannot be used with the –CAM option. If neither –DAM nor –CAM is specified, the subfiles of the restored segment directory assume the same file type as those of the source segment directory.
–NO_QUERY	Suppresses system questions during the restore operation.
–PROTECT	Gives the active file the same ACL protection as the archived file.
–REPORT	Reports each RBF subfile as it is restored. The default is not to report successful restores of individual subfiles. With or without the –REPORT option, REST_RBF reports the successful restore of the entire RBF.

For details on REST_RBF and on ROAM files, see the *ROAM Administrator's Guide*.

▶ RESUME pathname [arguments]

RESUME executes either a runfile or a CPL program.

pathname is a CPL program (whose filename must have a .CPL suffix) or a runfile. The runfile can be a dynamic-mode Executable Program Format (EPF) file linked with the BIND command or an R-mode runfile loaded with the LOAD command, but not a static-mode runfile loaded with the SEG command.

RESUME searches for file suffixes in the following order:

1. .RUN
2. .SAVE
3. .CPL
4. No suffix

The definition of **arguments** depends on whether you are executing a CPL program or a runfile.

Note

Do not use RESUME to execute a file with a .SEG suffix. Use the SEG command instead.

Executing EPF Runfiles

EPF runfiles are created by the BIND linking utility and have .RUN suffixes. These dynamic-mode runfiles are loaded into memory by PRIMOS, which allocates address space at execution time. Use the following format to execute an EPF runfile:

RESUME pathname [program-arguments]

pathname is the name of the EPF runfile. You do not need to specify the .RUN suffix because PRIMOS looks for it first. For example, to execute an EPF named PROG1.RUN which is in your current directory, issue the command

```
OK, RESUME PROG1
```

program-arguments are arguments passed to the program.

Executing R-mode Runfiles

R-mode runfiles are static-mode files with .SAVE suffixes or without suffixes. Files without suffixes are executed as .SAVE files. R-mode runfiles are created with the LOAD command.

R-mode runfiles are loaded into memory, using the saved RVEC values. You cannot change the SA and EA values but you can change one or more of the other five values by specifying them in **arguments** with the following format:

RESUME pathname [pc] [a] [b] [x] [keys] [program-arguments]

See Appendix A for an explanation of RVEC parameters and how you can specify new values. **program-arguments** are nonnumeric arguments passed to the program.

Executing CPL Programs

CPL programs have .CPL suffixes. To execute a CPL program, use the format

RESUME pathname [CPL-arguments]

CPL-arguments are passed to the program as CPL arguments. You can also use the CPL, JOB, and PHANTOM commands to execute a CPL program. For more information on CPL programs, see the *CPL User's Guide*.

▶ RESUS option

Operator Command

RESUS invokes the Remote System User facility, which allows remote operation of the supervisor terminal from any user terminal. RESUS is used with the Distributed System Management (DSM) facility.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ REVERT_PASSWORD

REVERT_PASSWORD converts the current directory from an ACL-protected directory to a password-protected directory.

Both of the following conditions must exist before you can convert an ACL directory:

- You must have Protect (P) access rights to the directory.
- The directory cannot have any access categories or subdirectories protected by ACLs.

If the directory was initially created as a password directory, its original password and protection keys are restored. If the directory was not previously a password directory, REVERT_PASSWORD sets the owner password as blank and the nonowner password as null. To change these defaults, use the PASSWD command.

For further details, see the *PRIMOS User's Guide*.

▶ RJE

RJE is not a PRIMOS command. Remote Job Entry (RJE) Phase II products are separately priced Prime software that enable multiuser Prime systems to emulate other vendor's RJE terminals over half-duplex, point-to-point, synchronous, and dialup or dedicated communications lines.

The two PRIMOS commands used with RJE are

- The RJQ command, which provides user interface to RJE
- The RJOP command, which provides operator control of RJE

The following terminals can be emulated with RJE:

- IBM 2780 and 3780
- HASP
- CDC 200UT
- Honeywell GRTS

- Univac 1004
- ICL 7020
- XBM (CO3)

For detailed information on the Prime Remote Job Entry emulators, see the *Remote Job Entry Phase II Guide*.

See also RJQ; RJOP.

▶ RJOP

Operator Command

RJOP is the Remote Job Entry (RJE) operator command that provides special RJE operational privileges to the RJE local site operator.

Before invoking RJOP, the operator must create a site definition file that contains information about the particular type of RJE terminal being emulated, the line type used, and other emulator attributes.

RJOP allows the operator to monitor and control the transmission and reception of files, and to send messages to a remote machine. Other commands provided through RJOP allow the manipulation and control of access to RJE file transmission queues. The operator can also use the RJQ command to queue files for transmission, list RJE queue entries, cancel unwanted entries from the queue, and restart aborted file transmissions. For detailed information on RJOP, see the *Remote Job Entry Phase II Guide*.

See also RJE; RJQ.

▶ RJQ

RJQ queues files for transmission to a remote site.

Remote Job Entry (RJE) users use the RJQ command to perform the following three functions:

- Queue a file for transmission to a remote site.
- Manage queued files. This function includes listing the RJE file transmission queue entries, canceling unwanted entries from the queue, and restarting aborted file transmissions.
- Build concatenated files.

The command format for RJQ differs according to the function you are performing. The formats are described in the following sections.

Queuing a File

To queue a file for transmission to a remote site, use the following RJQ command format:

RJQ **pathname** [–TO] **queuename [queue-options]**

pathname is the name of the file to be transmitted. You can specify a filename if the file is in your current directory. If you use the –NO_COPY option, you must specify the full pathname of the file.

queuename is the name of the queue in the remote system to which the file is being sent. The default name for **queuename** is the site name of the remote system. You do not have to specify the –TO keyword if **queuename** is the third item on the command line.

queue-options are listed below. You can specify more than one option if they do not conflict.

Queue Option	*Meaning*
–AS **internal-name**	Defines the name that appears in the name field when you issue the RJQ –LIST command. (Default is the file's pathname, without passwords.)
–DEFER **time**	Delays transmission of the file until the specified time. **time** is given in the 24-hour hh:mm format. (The colon is optional.)
–DELETE	Deletes the user's source file after the file has been successfully entered in the queue.
–DEVICE **device-name**	Specifies the device to which or from which the file is being sent. **device-name** is one of the following: CPn, CRn, or LPn, where **n** is a device number from 1 to 7, inclusive. In slave (terminal) mode, files can be sent only as card input (CR). In master (host) mode, files can be sent to a remote printer (LP) or a remote punch (CP). In HASP and XBM only, more than one of each type of device may be on the system (for example, LP3).
–KEEP_REQUEST	Prevents the emulator from deleting the queue entry after transmission, but marks the entry as "Sent" in the queue.
–NO_COPY	Prevents a temporary copy of the user data from being made and queued for transmission. If you use this option, you must specify a full pathname for the file.
–NO_TRANSLATE	Sends the file without translation.
–VFC **[argument]**	Defines the vertical forms translation codes to be sent with line printer files in master (host) mode. **argument** is either FTN or NONE (the default). FTN is supported only for 2780, 3780, or HASP emulators.

–WITH **protocol**	Enforces the file-sending protocol specified in the remote site's site definition file. **protocol** is one of the following: 2780, 3780, HASP, GRTS, 7020, XBM, 1004, or 200UT.

After the file is entered in the queue, it is assigned a queue entry number in the following format:

RJnnnn

nnnn is a four-digit number. RJ0004 is an example of a queue entry number.

Managing Queued Files

Use the following command format to manage files in the RJE file transmission queue:

RJQ { –LIST / –CANCEL / –RESET } **[entry-argument] [manage-options]**

The three keywords specify the type of control you want to impose on the queued entries, as follows:

- –LIST lists some or all queue entries.
- –CANCEL cancels some or all of your own queue entries.
- –RESET resets the state of one of your files that aborted during transmission, so that the file may be restarted and transmission retried.

entry-argument specifies the entry or entries to be acted upon and is one of the following:

Entry Argument	*Meaning*
entry-number	Acts only on the queue entry specified by **entry-number**. You can specify **entry-number** either as a full queue entry number (RJnnnn, where **nnnn** is a four-digit number) or as the four-digit number that follows the RJ prefix. You can specify more than one number if you separate each one with a comma.
ALL	Acts on all entries in the general queue of submitted file transmission requests. (Default for –LIST.) Users cannot specify ALL for –CANCEL or –RESET. The operator (that is, a user logged in as SYSTEM) can use ALL with –CANCEL to cancel all entries in the referenced queue.
OWN	Acts on all the specific queue entries for file transmission requests submitted under your login ID.

manage-options is one or more of the following:

Manage Option	*Meaning*
–DEFER **time**	Lists only entries deferred beyond the specified time. Can be used only with –LIST. **time** is given in the 24-hour hh:mm format. (The colon is optional.) The default for **time** is the current time.
–TO **queuename**	Acts only on entries in the specified queue.
–WITH **protocol**	Acts only on entries queued with the given protocol. The legal arguments for **protocol** are 2780, 3780, HASP, GRTS, 7020, XBM, 1004, and 200UT.

Building Concatenated Files

RJQ has an interactive mode that allows you to build concatenated files. To enter this mode, issue the RJQ command without any arguments or options. A right angle-bracket (>) prompt preceded by "Concat." is displayed, as follows:

```
OK, RJQ
Concat. >
```

You must enter a RJQ subcommand to continue. Enter HELP to invoke the RJQ Help facility.

For complete information on RJQ, see the *Remote Job Entry Phase II User's Guide*.

▶ RLS

See RELEASE_LEVEL.

▶ ROSAU

Operator Command

ROSAU invokes the ROAM System Administrator Utility (ROSAU).

The ROSAU command can be issued only by ROAM Administrators and allows them to perform the following functions:

- Management of ROAM system files
- Management and execution of after-image recovery
- Fixing of system and user files in special circumstances

When the utility is invoked, it displays a right angle-bracket (>) prompt and waits for you to enter a ROSAU command. For complete information on the operation and commands of ROSAU, see the *ROAM Administrator's Guide*.

▶ RSTERM [–READ] [–WRITE]

RSTERM empties the user terminal's read (input) and/or write (output) buffers.

Use RSTERM when you want to discard either typed-ahead input or output awaiting transmission to your CRT. The command is useful in CPL programs for reinitializing the terminal state when a condition handler for the QUIT$ condition is specified.

Specifying –READ empties the input buffer. Specifying –WRITE empties the output buffer. Specifying neither option empties both buffers.

RSTERM is also supported over the Network Terminal Service (NTS) and empties both the LAN300 Host Controller (LHC300) and the LAN300 Terminal Server (LTS300) input and/or output buffers, as well as the PRIMOS buffers.

▶ RUNOFF [pathname]

RUNOFF invokes Prime's text formatting program, RUNOFF.

pathname specifies the ASCII source file that is to be formatted. If you do not specify **pathname**, RUNOFF prompts you for its name, as follows:

```
OK, RUNOFF
[RUNOFF Rev. 21.0 Copyright (c) Prime Computer, Inc. 1985]
INPUT FILE:
```

The input file must be in RUNOFF input form. It normally contains the necessary RUNOFF commands to control margins, indentation, line spacing, running heads, page numbering, and other textual matters.

After you have specified the input filename (either on the command line or at the `INPUT FILE:` prompt), RUNOFF displays a dollar-sign ($) prompt so that you can enter any further RUNOFF commands. To finish entering RUNOFF commands, enter a blank line (a carriage return without text).

RUNOFF produces a formatted output to the terminal or to a designated disk file. For detailed and tutorial information, see the *New User's Guide to EDITOR and RUNOFF.*

▶ RWLOCK pathname lock [–REPORT]

RWLOCK sets the read/write concurrency lock on a file or segment directory.

The setting of a concurrency lock determines how many readers and/or writers can access the file or segment directory at one time. **pathname**, which can be a wildcard name, identifies the object to be protected. You must have Owner access to the object if it is a password directory, or Protect (P) access if it is an ACL directory.

lock specifies the setting of the read/write concurrency lock. The value of **lock** must be only one of the following:

Lock	*Setting*
SYS	Sets protection to the value of the system read/write lock. (Default) The system lock is determined by the System Administrator.
EXCL	Sets for **n** readers *or* one writer (exclusive OR). An unlimited number of readers or one writer can access the object simultaneously.
UPDT	Sets for **n** readers *and* one writer. An unlimited number of readers and one writer can access the object simultaneously.
NONE	Sets for **n** readers *and* N writers. All users can access the object simultaneously.

The –REPORT option reports the result of each successful lock setting.

The following example illustrates the RWLOCK command:

```
OK, rwlock mail.list excl -report
"MAIL.LIST" locked.
OK,
```

▶ S

See START.

▶ SAC

See SET_ACCESS.

▶ SAVE pathname [sa] [ea] [pc] [a] [b] [x] [keys]

SAVE saves the contents of memory (segment 4000_8) as a diskfile named **pathname**.

Because SAVE writes the contents only of segment 4000_8, use SAVE only to save static-mode programs residing entirely in segment 4000_8. You cannot use SAVE on multisegment programs, which do not reside entirely in segment 4000_8 or on EPFs, which never reside in segment 4000_8.

The contents to be saved begin at **sa** (starting address) and end at **ea** (ending address). The other arguments are the RVEC parameters described in Appendix A of this guide. If you do not specify new values for these parameters, SAVE takes the existing values of RVEC from the current register set and stores them with the program. The RVEC parameters are used to initialize the processor registers and keys when the program is restored or resumed.

The SAVE command produces only static-mode runfiles (.SAVE files). To produce 32I or 64V segmented runfiles (.SEG files), use the SEG command. To produce EPF runfiles (.RUN files), use the BIND command.

Note

All FORTRAN programs begin with ELM (Enter Load Mode). If macro assembler (PMA) users have ELM as the first instruction in the program, they do not have to use the SAVE command to set the keys after loading. The preferred way to save a memory image is to use the SAVE subcommand of LOAD, which automatically sets up the program keys as appropriate.

▶ SAVE_RBF source-pathname target-pathname [options]

SAVE_RBF archives a ROAM file to disk.

Before you can use SAVE_RBF, the after-image file must be on the system containing the ROAM file to be stored. **source-pathname** is the name of a ROAM master segment directory. **target-pathname** is the name of the archived master file and cannot be an existing filename.

SAVE_RBF has the following options:

Option	*Meaning*
–CAM	Archives all the subfiles of the target segment directory as contiguous files. Cannot be used with the –DAM option. If neither –CAM nor –DAM is specified, the subfiles of the archived segment directory assume the same file type as those of the source segment directory.
–DAM	Archives all the subfiles of the target segment directory as DAM files. Cannot be used with the –CAM option. If neither –DAM nor –CAM is specified, the subfiles of the archived segment directory assume the same file type as those of the source segment directory.
–PROTECT	Gives the archived file the same ACL protection as the source file.
–REPORT	Reports each RBF subfile as it is archived. The default is not to report successful saves of individual subfiles. With or without the –REPORT option, SAVE_RBF reports the successful save of the entire RBF.

To archive an active or inactive ROAM file to magnetic tape, use the BRMS or MAGSAV utilities. For details on SAVE_RBF and on ROAM files, see the *ROAM Administrator's Guide*.

▶ SCHDEC [schema-name] [output-pathname]

SCHDEC invokes the DBMS Schema Decompiler (SCHDEC).

The schema decompiler translates a compiled schema back to an ASCII file (that is, into its DDL description). If you omit **schema-name** and **output-pathname**, SCHDEC prompts you for them. Access to the schema decompiler is limited to Database Administrators. For detailed information, see the *DBMS Administrator's Guide*.

▶ SCHED [schema-name]

SCHED invokes the DBMS Schema Editor (SCHED). The schema editor is an interactive processor that allows a Database Administrator to alter the definition of a database. If you do not specify a schema name, SCHED prompts you for one. For details, see the *DBMS Administrator's Guide*.

▶ SCHEMA source-pathname [–OUTPUT pathname] [–LIST pathname] [–DAM]

SCHEMA invokes the DBMS schema Data Description Language (DDL) compiler.

The DDL compiler translates **source-pathname** (which contains the schema definition) into the schema table.

The schema table is placed in your current directory unless you use the –OUTPUT option to place it in another directory.

The –LIST option specifies an alternative location or name for the output listing. The default name and location is **filename**.LIST in the directory of the source file.

The –DAM option causes subfiles to be created as DAM files. Without the –DAM option, subfiles are created as contiguous (CAM) files.

For more information on SCHEMA, see the *DBMS User's Guide* or the *DBMS Data Description Language Reference Guide*. For more information on the –DAM option, see the *DBMS Administrator's Guide*.

▶ SE

See SETIME.

▶ SECURITY_MONITOR [options]

Operator Command

SECURITY_MONITOR activates or deactivates the Audit Collection facility and allows the selection of the type(s) of system events to be monitored. SECURITY_MONITOR may be issued only from the supervisor terminal or by a user logged in as the System Administrator. For more information, see the *System Administrator's Guide, Volume III: System Access and Security.*

▶ SECURITY_STATUS [options]

Operator Command

SECURITY_STATUS displays the current status of the Audit Collection facility. SECURITY_STATUS may be issued only from the supervisor terminal or by a user logged in as the System Administrator. For more information, see the *System Administrator's Guide, Volume III: System Access and Security.*

▶ SEG [{ pathname / –LOAD }]

SEG invokes a utility for loading, modifying, running, and sharing segmented (V-mode and I-mode) programs.

Executing Runfiles

Segment directories (also called SEG runfiles) are static V-mode and I-mode runfiles. To execute a SEG runfile, use the following command format:

SEG pathname

pathname is the name of the runfile.

Creating SEG Runfiles

To load (that is, create) a V-mode runfile from an object (binary) file, use the following format:

SEG –LOAD

This command invokes the SEG utility's LOAD subprocessor. The subprocessor displays a dollar-sign ($) prompt and waits for a LOAD subcommand that loads an object file whose name usually has a .BIN suffix. For example,

```
OK, SEG -LOAD
[SEG Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
$
```

The runfile generated by the LOAD subprocessor uses the object file's basename and adds a .SEG suffix to produce the name of the output runfile.

If you issue SEG without a pathname or without the –LOAD option, the utility enters SEG command level and displays a number-sign (#) prompt, as follows:

```
OK, SEG
[SEG Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
#
```

You must enter a SEG subcommand to continue the loading procedure. To display a list of SEG subcommands, enter HELP at the # prompt.

Assignment of Static Program Segments

PRIMOS assigns static segments to you as the segments are accessed. The segments are not reassigned until you issue the LOGOUT or the DELSEG command. The maximum number of segments available to a user program is between 32 and 256. The number is configured by the System Administrator and is available from the administrator.

User segments start at segment 2048 (4000_8). Segments in other ranges are for PRIMOS itself, for shared code, or reserved for expansion. For information on installation of shared code, see the *System Administrator's Guide, Volume I: System Configuration.*

For a complete discussion of SEG, see the *LOAD and SEG Reference Guide.*

▶ SETIME –mmddyy –hhmm

Operator Command

SETIME sets the date and time of day for the system.

mmddyy is a six-digit number that represents the month, day, and last two digits of the year. **hhmm** is a four-digit number that represents the hours and minutes in 24-hour notation. Both numbers must be preceded by a hyphen. For details, see the *Operator's Guide to System Commands.*

▶ SETMOD { –USER | –OPERATOR | –NOASSIGN }

Operator Command

SETMOD determines how magnetic tape drives are assigned.

SETMOD –USER (the default) allows users to assign tape drives from their terminals with the ASSIGN command.

SETMOD –OPERATOR requires operator intervention. When a user issues the ASSIGN command to request a tape drive, the request is displayed at the supervisor terminal. The operator responds to the request and notifies the user (via a message at the user's terminal) of the result.

SETMOD –NOASSIGN prohibits all tape drive assignments. In this mode, use of the ASSIGN command produces a message at the user's terminal saying that tape drives cannot be assigned.

For further information, see the *Operator's System Overview* and the *Operator's Guide to System Commands.*

▶ SET_ACCESS target-pathname [{ acl [–NO_QUERY] | –LIKE reference-name | –CATEGORY categoryname }]

SET_ACCESS specifies the access rights for a file, directory, or segment directory.

target-pathname specifies the object (file, directory, or segment directory, but not MFD) for which access rights are to be set. The access rights for the object are defined in an Access Control List (ACL).

Note

Use the SET_ACCESS command only to create a new ACL. To modify (edit) an existing ACL, use the EDIT_ACCESS command.

Access Control Lists

An ACL is a list of users and the access rights granted to each user. The list may contain a maximum of 32 entries, but the list cannot exceed 160 characters, including blanks.

The entries in an ACL define the rights that a user has when accessing a file or directory. Each entry is an ordered pair, in the following format:

identifier:rights

identifier is one of the following:

- A user ID that identifies a single user.
- A group name that identifies a set of users who are members of a particular access group. Your System or Project Administrator defines access groups for your system or project. The group name must begin with a period (.).
- The special identifier $REST that identifies all users who are not listed by user ID or are not members of a group.

rights identify the privileges given to a user when using a particular file or directory. Specify **rights** by using one or more of the mnemonic codes listed below:

Code	*Right*	*Applies to*	*Allows the user to*
P	Protect	Directories	Change accesses and attributes.
D	Delete	Directories	Delete directory entries.
A	Add	Directories	Add directory entries.
L	List	Directories	Read directory contents.
U	Use	Directories	Attach to directory.
R	Read	Files	Read file contents.
W	Write	Files	Change file contents.
X	Execute	Local EPFs	Execute, but not read or copy local EPF.
O	Owner	Directories and files	Set read/write locks without P or ALL access rights.
ALL	All	Directories and files	All of the above.
NONE	None	Directories and files	All access denied.

Note

Read (R) access includes Execute (X) access. Therefore, users who already have R access to a file do not need X access as well.

When you list the contents of an ACL with the LIST_ACCESS command, the display at the terminal resembles the following example:

```
OK, LIST_ACCESS MEMO

"MEMO" protected by default ACL (from "<MARKET>JULIE"):
        JULIE:      ALL
        .SALES:     DALURW
        $REST:      NONE
OK,
```

Types of ACLs

There are two types of ACLs: specific ACLs and access categories.

A **specific ACL** is an unnamed attribute of a file or directory. Specific ACLs do not exist as separate, named file system objects but are linked to the objects they protect. A specific ACL, therefore, does not appear when you issue the LD command.

An **access category** is a named file system object that has a separate existence and resides in a directory. The entryname of an access category has the suffix .ACAT (for example, GUARD.ACAT). The LD command lists access categories.

Each access category contains an ACL. The access category may protect any number of files, directories, and segment directories in the directory in which it resides. Thus, access categories provide an efficient way to group objects together for access control purposes.

Using SET_ACCESS

There are four SET_ACCESS command formats: one for each of the three options and one that uses only a target pathname. Each format is described below. To use SET_ACCESS, you must have Protect (P) rights to the targeted object.

To use SET_ACCESS on your current directory (that is, the directory to which you are currently attached), use the full pathname of the directory. For example, if the pathname of your current directory is <CITY>HOUSE>ROOM, use the command line

```
OK, SET_ACCESS HOUSE>ROOM MARY:ALL $REST:LUR
```

If your current directory is a top-level directory (for example, <CITY>HOUSE), use the diskname in the target pathname, as follows:

```
OK, SET_ACCESS <CITY>HOUSE MARY:ALL $REST:LUR
```

If the target is a password directory whose parent is an ACL directory, the command converts the password directory to an ACL directory.

Specifying the ACL Option

To create an ACL for an object, use the format

SET_ACCESS target-pathname acl [–NO_QUERY]

target-pathname is a file, directory, or access category. **acl** is a list of one or more identifiers and rights in the following format:

identifier-1 :rights-1 [... identifier-n :rights-n]

identifier is a user ID, a group name, or $REST. **rights** is one or more of the mnemonic protection codes listed in the section above, Access Control Lists. **identifier** and **rights** must be separated by a colon (:) but not by any blanks. Each set of **identifier:rights** must be separated from the next by at least one blank.

You can specify a maximum of 32 sets of identifiers and rights, but the total number of characters of all the sets cannot exceed 160. For further information on identifiers and rights, see the section above, Access Control Lists.

The result of the ACL option depends on the target's type and current protection. The possibilities are

- If the target is a file or directory protected by neither a specific ACL nor an access category, SET_ACCESS creates a specific ACL with the specified identifiers and rights.
- If the target has an existing specific ACL, SET_ACCESS asks you if it is OK to replace the old ACL with the new one. Use the –NO_QUERY option to suppress this question.
- If the target is protected by an access category but not by a specific ACL, SET_ACCESS removes the target from the category and creates a new specific ACL to protect the target. The contents of the category, however, are not changed. The target is henceforth protected by the specific ACL.
- If the target is an access category, SET_ACCESS asks you if it is OK to replace its current contents with the new ACL you specified. Use the –NO_QUERY option to suppress this question.
- If the target does not exist, SET_ACCESS creates it as an access category with the name **target-pathname**.ACAT. (The .ACAT suffix is added automatically.) The category's contents are the sets of identifiers and rights you specified.

Note

A user who is in more than one group has the sum (logical union) of all access rights for those groups. If a user has access rights from both a user ID and a group name, the user receives only those rights assigned by the user ID.

Example 1: In this example, user SAWYER creates a specific ACL for the current directory CAVE (which is a subdirectory of the directory SAWYER), and then uses the LIST_ACCESS command to check the ACL.

```
OK, SAC SAWYER>CAVE SAWYER:ALL FINN:DALURW .GANG:LUR $REST:NONE
OK, LAC SAWYER>CAVE

ACL protecting "SAWYER>CAVE":
        SAWYER:   ALL
        FINN:     DALURW
        .GANG:    LUR
        $REST:    NONE
```

User SAWYER now has ALL access rights to CAVE; user FINN has all rights except Protect (P) and Owner (O); members of the user group named .GANG have List, Use, and Read access; and all other users have no access rights of any kind (NONE).

Example 2: In this example, user SAWYER sets access rights on a file named FENCE that already has a specific ACL:

```
OK, SET_ACCESS FENCE SAWYER:ALL $REST:LUR
A specific ACL for "FENCE" already exists.
Do you want to replace it? YES
OK, LIST_ACCESS FENCE

ACL protecting "FENCE":
        SAWYER:   ALL
        $REST:    LUR
OK,
```

SAWYER now has ALL access rights to FENCE while all other users have List, Use, and Read access only.

Example 3: This example illustrates how to create an access category:

```
OK, SET_ACCESS GUARD SAWYER:ALL .GANG:LUR $REST:NONE
"GUARD.ACAT" does not exist.  Create access category? YES
OK,
```

Issuing the LD command would show that an access category named GUARD.ACAT was in the directory. The ACL in GUARD.ACAT gives SAWYER ALL access rights. Members of .GANG have List, Use, and Read rights, while other users have no access rights (NONE). Because GUARD.ACAT is newly created, it does not yet protect any objects in the directory. Use the –CATEGORY option of SET_ACCESS, described below, to protect an object with an access category.

Specifying the –LIKE Option

The –LIKE option sets the same protection on a target as that set on another file system object. The format is

SET_ACCESS **target-pathname** –LIKE **reference-name**

target-pathname is an existing file or directory that is to receive the same protection that **reference-name** has. **reference-name** must be an existing file, directory, or access category. A specific ACL is created for the target if it does not have one.

The ACL of the target is set identically to that of the reference. If the target is protected by an access category, the target is removed from the category and the category is not otherwise changed.

In the following example, user SAWYER sets the same protection on the file RAFT as that possessed by the file FENCE (from the previous example):

```
OK, SET_ACCESS RAFT -LIKE FENCE
OK, LIST_ACCESS RAFT

ACL protecting "RAFT":
        SAWYER:  ALL
        $REST:   LUR
OK,
```

The files FENCE and RAFT now have specific ACLs with identical contents.

Specifying the –CATEGORY Option

The –CATEGORY option adds a file or directory to the list of objects protected by an access category. The format is

SET_ACCESS **target-pathname** –CATEGORY **categoryname**

categoryname is an existing access category that is to provide protection for **target-pathname**.

For example, to protect the file RIVER with the existing access category named GUARD.ACAT, user SAWYER issues the following command:

```
OK, SET_ACCESS RIVER -CATEGORY GUARD
OK, LIST_ACCESS RIVER

ACL protecting "RIVER" (from access category
"<TWAIN>SAWYER>GUARD.ACAT"):
        SAWYER:  ALL
        .GANG:   LUR
        $REST:   NONE
OK,
```

Specifying No Options

Use the SET_ACCESS command with only a target pathname to set an object's protection to the default access rights for the parent directory. This command format is

SET_ACCESS target-pathname

target-pathname must be a file, directory, or segment directory, but not an MFD. The following example illustrates this form of SET_ACCESS:

```
SET_ACCESS STAT.REPORT
```

Whether STAT.REPORT is a file, segment directory, or subdirectory, it acquires the default protection granted by its parent directory.

▶ SET_ASYNC { –DISPLAY [–LINE n] | –LINE n [–TO m] [options] | –HELP }

SET_ASYNC, as a user command, allows you to configure an asynchronous line (assigned to you) that is connected to an AMLC, ICS, or LTS controller. As an operator command, SET_ASYNC allows you to control certain administrative characteristics of any asynchronous line. SET_ASYNC replaces the AMLC command.

SET_ASYNC allows the nonprivileged user to set assignable line characteristics, or to set terminal line characteristics, including Network Terminal Service (NTS) line characteristics. The user options for specifying these characteristics are listed below.

Setting terminal line characteristics with SET_ASYNC is effective only for the current terminal session. Setting assignable line characteristics with SET_ASYNC is effective until those charcteristics are changed again with SET_ASYNC for the specified lines.

Main SET_ASYNC User Options

The following list describes the main SET_ASYNC options for the nonprivileged user.

Option	*Meaning*
–DISPLAY	Displays all line characteristics for the specified line.
–HELP	Displays command syntax and a complete list of available options.

–LINE **n**	Specifies the decimal line number to be configured. **n** is expressed in *decimal* notation.
–TO **m**	Configures a range of consecutively numbered lines with identical options. **m** specifies the last line number in a series beginning at the line number **n** specified in the –LINE option. The value of **m** must be greater than **n**.

SET_ASYNC –LINE Options

The options below are used to specify the characteristics of one or more than one asynchronous line. Enter these options after the –LINE option on the command line. These options replace the octal parameters used by the obsolete AMLC command. To find out the SET_ASYNC equivalent of a given AMLC command, use the interactive form of the CONVERT_AMLC_COMMANDS utility.

Option	*Meaning*
–DEFAULT	Sets all options to their default setting, as shown in Table 2-3. Desired exceptions to default settings may be specified after –DEFAULT on the command line.
–ECHO	Sets the line to full duplex (default).
–NO_ECHO	Sets the line to half duplex.
–PROTOCOL **name**	Defines the format and relative timing of data. **name** may be any of the following. (TTY is the default.)

ASD	TTY8
TRAN	TTY8HS
TT8BIT	TTYNOP
TTY	TTYUPC

NTS lines support only TTY, TRAN, TTY8BIT, and TTYUPC; do not use ASD or TTYNOP on an NTS line.

The obsolete protocols TTYHS, TRANHS, and TTYHUP are supported for older model 5054 AMLC controller boards.

Note

If you select the TRAN protocol, all characters normally pass transparently as data. However, if you select the SET_ASYNC –XOFF option or if you set **lword** to bit 3 (recognize XOFF) with the AMLC command, then PRIMOS recognizes XON and XOFF as flow control characters, even if TRAN is in effect.

For descriptions of standard protocols, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

–SPEED value

Sets the baud rate for the asynchronous line. **value** may be any of the following:

50	200	2400	19200
75	300	3600	CLOCK (2)
110	600	4800	SPEEDA (3)
134.5	1200 (1)	7200	SPEEDB (3)
150	1800	9600	SPEEDC (3)

(1) Supplied with the –DEFAULT option
(2) Speed set with AMLCLK directive
(3) Three jumper speeds set with the ASYNC JUMPER directive.

For further information on the ASYNC JUMPER and AMLCLK directives, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

–STOP_BITS n

Signals the receiving device to wait for the next character. **n** defines the number of stop bits to use, either 1 (default) or 2. All characters have 1 start bit, 1 parity bit, 7 information bits, and either 1 or 2 stop bits. The slower the transmission speed, the more stop bits are required to recognize the EOT. For this reason, an 11-bit character length and two stop bits are used for devices that operate at 110 baud.

–PARITY value

Appends parity bits from the transmitting device to characters. –PARITY is used for error detection. **value** can be ODD, EVEN, or NONE (default). This option either sets the line parity to the desired setting or disables parity.

–CHAR_LENGTH n

Sets the number of information and parity bits per character. **n** can be 5, 6, 7, or 8 (default). Character length can be adjusted for the nonstandard character sizes required by Baudot terminals, telex lines, or foreign devices. PRIMOS right-justifies the bits in a byte and sets the leftmost bits to zero. For more information, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

–LINE_FEED

Echos a line feed character for the RETURN key. This option is valid only when –NO_ECHO (half duplex) is specified. –NO_LINE_FEED is the default.

–NO_LINE_FEED

Does not echo a line feed character for RETURN. (Default)

–XOFF

Enables CONTROL-S and CONTROL-Q (–XON and –XOFF) to respectively stop and start the flow of data on the line from the host to the terminal. (Default)

-NO_XOFF	Disables CONTROL-S and CONTROL-Q. This option is used for devices that cannot recognize these control key sequences.
-REVERSE_XOFF	Enables Reverse Flow Control (RFC) for asynchronous lines. RFC sends XOFF characters to a device when the PRIMOS input ring buffer is 60% full. When the input ring buffer drops to 20% full, an XON character is sent to the device to indicate that transmission can resume. RFC also attempts to prevent DMQ input queue overruns for all ICS3 controllers and any ICS2 controllers that are not using BSC and ASYNC protocols. Choose this option only for lines connected to devices that can interpret XON and XOFF characters (such as PT45, PST 100, and PT200 terminals).
-NO_REVERSE_XOFF	Disables Reverse Flow Control for the line. (Default)
-DATA_SENSE_ENABLE	Enables the Data Set Sense (DSS) protocol, (reverse channel). This option is used for transmitting control information or for controlling the flow of data to devices that do not recognize XON/XOFF.
-NO_DATA_SENSE_ENABLE	Disables the Data Set Sense (DSS) protocol (reverse channel). (Default)
-DATA_SET_SENSE **ready_value**	Supports devices that toggle an RS-232-C pin (usually pin 8) to indicate when they are busy/ready instead of using XON/XOFF. The Data Set Sense (DSS) protocol sets **ready_value** as either HIGH (pin signal raised) or LOW (pin signal lowered). The default is HIGH. Some devices use pins other than pin 8. If this is the case, ask your Customer Support Center to arrange your cables so that the Data set sense signal is wired into the pin used for carrier detect. If you use the -DSS option, you must also use -DSE and -NO_DSE to enable and disable flow control. You must specify -NO_XOFF as well. Data Set Sense is also referred to as buffered protocol or reverse channel protocol.
-DATA_SET_CONTROL	Required for modems and port selectors to recognize when a block of information is transmitted. This option is ignored by terminals. Do not use -DSC on NTS lines. (Default)
-NO_DATA_SET_CONTROL	Disables the -DSC option.
-ERROR_DETECTION	Used only for testing. When an input buffer overflows or when a parity error is detected, the incoming character is replaced with an ASCII NAK.
-NO_ERROR_DETECTION	Prevents the line from sending an ASCII NAK character if an input parity or input buffer overflow error is sensed. (Default)

–SYSTEM Sets the line characteristics for the current line to system default settings that were in effect when you logged in.

Note

You can make changes only to your terminal line or to lines that you have assigned with the ASSIGN command. As a nonprivileged user, you cannot issue the SET_ASYNC options –ASSIGNABLE, –SPEED_DETECT, –NO_SPEED_DETECT, –LOOP_LINE, –NO_LOOP_LINE, –DISLOG, –NO_DISLOG, or USER_NUMBER. For more information on the SET_ASYNC command, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers.*

Example of SET_ASYNC

The following example illustrates the use of SET_ASYNC to disable CONTROL-S and CONTROL-Q (–XON and–XOFF):

```
OK, SET_ASYNC -DISPLAY
[Set_Async Rev 21.0 (c) Prime Computer, Inc. 1986]

LINE = 0
     PARity          = NONE     ECHO
     PROtocol        = TTY      XOFF
     SPEED           = 9600     NO_LOOP
     Stop_Bits       = 1        Line_Feed
     Char_Length     = 8        NO_Data_Sense_Enable
     REVerse_XOFF    = OFF      NO_ERRor_DETection
     ASsiGNable      = NO       NO_Speed_Detect
     Data_Set_Sense  = LOW      DISLOG
     Owner Process   = 2        Data_Set_Control

OK, SET_ASYNC -LINE 0 -NOXOFF
[Set_Async Rev 21.0 (c) Prime Computer, Inc. 1986]
OK, SET_ASYNC -DISPLAY
[Set_Async Rev 21.0 (c) Prime Computer, Inc. 1986]

LINE = 0
     PARity          = NONE     ECHO
     PROtocol        = TTY      NO_XOFF
     SPEED           = 9600     NO_LOOP
     Stop_Bits       = 1        Line_Feed
     Char_Length     = 8        NO_Data_Sense_Enable
     REVerse_XOFF    = OFF      NO_ERRor_DETection
     ASsiGNable      = NO       NO_Speed_Detect
     Data_Set_Sense  = LOW      DISLOG
     Owner Process   = 2        Data_Set_Control

OK,
```

TABLE 2-3
Default Settings for Asynchronous Lines Provided by the SET_ASYNC –DEFAULT Option

Option	*Setting*
–ASSIGNABLE NO	Line is a login line.
–CHAR_LENGTH 8	Character length is 8 bits.
–DATA_SET_CONTROL	Enables modems and port selectors to recognize when information is being transmitted.
–ECHO	Full duplex line.
–LINE_FEED	Echo LINE FEED and carriage return for each RETURN.
–NO_DATA_SENSE_ENABLE	Data Set Sense is disabled.
–NO_DISLOG	Disables automatic disconnect.
–NO_ERROR_DETECTION	NAK character is not placed in the input buffer when an input parity or input buffer overflow is detected.
–NO_LOOP_LINE	Line is not in loopback mode.
–NO_REVERSE_XOFF	Reverse Flow Control is not enabled.
–NO_SPEED_DETECT	Disables Auto Speed Detect.
–PARITY NONE	Line parity is disabled.
–PROTOCOL TTY	Line uses the terminal protocol.
–SPEED 1200	Line speed is 1200 bits per second.
–STOP_BITS 1	One stop bit.
–USER_NUMBER (n + 2)	Associates the buffers for USER_NUMBER (**n** + **2**) with the physical line number **n** specified in the –LINE **n** option. For example, the default buffer associated with –LINE 16 is buffer 18.
–XOFF	CONTROL-S stops and CONTROL-Q resumes the flow of data to the terminal.

▶ SET_DELETE pathname [{ –PROTECT / –NO_PROTECT }]

SET_DELETE sets the delete switch on a file system object.

When you create a file or a segment directory, its delete switch is set to allow you to delete it without your verification. The SET_DELETE command changes this switch

so that PRIMOS must first ask for your verification before it can delete the object. This protection, which is similar to the default protection on directories, assures that you cannot unintentionally delete the object.

SET_DELETE works only for objects in ACL-protected directories. Before you can use this command, you must have Delete (D) access to the directory that contains the object.

pathname is the file, directory, or segment directory to receive delete-protection. Access categories cannot be protected. When you use this command on directories and segment directories, they receive double protection because they still retain their original default protection.

The –PROTECT option, which is the default, sets the protection on **pathname**. The –NO_PROTECT option removes the delete-protection from **pathname**.

Example 1: This example illustrates both SET_ACCESS options by first setting delete protection on the file SPECS and then removing it:

```
OK, SET_DELETE SPECS -PROTECT
OK, DELETE SPECS
"SPECS" protected, ok to force delete? NO
OK, SET_DELETE SPECS -NO_PROTECT
OK, DELETE SPECS
OK,
```

Example 2: This example shows how PRIMOS asks for your verification twice when you delete a directory that is delete-protected:

```
OK, CREATE SUBDIR.3
OK, SET_DELETE SUBDIR.3 -PROTECT
OK, DELETE SUBDIR.3 -REPORT
Ok to delete directory "SUBDIR.3"? YES
"SUBDIR.3" protected, ok to force delete? YES
"SUBDIR.3" deleted.
OK,
```

If you answer N or NO (in uppercase or lowercase) at either query, the directory is not deleted.

▶ SET_PRIORITY_ACCESS diskname acl

Operator Command

SET_PRIORITY_ACCESS sets a priority Access Control List (ACL) on a specified disk partition.

diskname is the name of the disk partition to be affected. **acl** is a list of identifiers and their access rights. If a SREST identifier is given as part of **acl**, the access rights specified by that identifier override any other access control in effect on the partition.

For detailed information on priority ACLs and the SET_PRIORITY_ACCESS command, see the *System Administrator's Guide, Volume III: System Access and Security*, or the *Operator's Guide to System Commands*.

▶ SET_QUOTA directory-pathname [–MAX n]

SET_QUOTA sets the quota on a directory to a specific number of records.

A **quota** is the maximum number of records that a directory can contain. At most sites, only the System Administrator can set quotas on top-level directories. With SET_QUOTA, users can set quotas on their own subdirectories, usually as a check on their own storage use.

To set a quota on a directory or subdirectory, you must have either Protect (P) access (for access control systems) or Owner access (for password systems) on the parent directory. If you use SET_QUOTA without having these rights, the quota is not set and the following error message is returned:

```
Insufficient access rights. directory-name (set_quota)
```

directory-pathname is the directory or subdirectory on which you are setting a quota. If you are setting a quota on a subdirectory within your current directory, you need to specify only the subdirectory's entryname (the final element of its pathname).

The –MAX option specifies the maximum number of records to be allocated to the directory or subdirectory. **n** is a positive decimal integer between 0 and the maximum number of records on the disk partition. If **n** is 0, no quota is set and the directory can use as many records as it can obtain. Omitting the –MAX option removes a quota, and is therefore the same as specifying –MAX 0.

The following example sets a quota of 1000 records on the subdirectory LETTERS and then uses the LIST_QUOTA command to verify the results:

```
OK, SET_QUOTA *>LETTERS -MAX 1000
OK, LIST_QUOTA *>LETTERS

Maximum records allowed on "*>LETTERS" = 1000.
Total records used = 1.
Records used in this directory = 1.
OK,
```

After you have set a quota on a directory or subdirectory, you can raise, lower, or remove the quota by issuing another SET_QUOTA command with a different value specified with the –MAX option.

▶ SET_RBF pathname [options]

SET_RBF sets or lists the attributes of a ROAM file.

The attributes of a ROAM file are the recovery and concurrency attributes of a transactional file and the locking attribute that allows exclusive offline use.

pathname is the name (full pathname or filename) of the master segment directory for which attributes are to be set or listed.

To set the attributes of **pathname**, specify one or more of the options listed below. If you do not specify an option, SET_RBF lists the file's existing attributes.

Option	*Meaning*
–AICHK	Specifies a runtime check of the save date/time stamp when after-imaging is enabled for the file. This option is meaningful only if the check was previously disabled.
–AIRCV	Turns on after-image recovery and enables before-image recovery.
–BIRCV	Enables the force-writing of before-images for before-image recovery.
–LOCK	Restricts access to the file. A locked file cannot be accessed by any run-units and can be unlocked only by the ROAM Administrator who issued the lock.
–NO_AICHK	Disables the runtime check of the save date/time stamp.
–NO_AIRCV	Turns off after-image recovery.
–NO_BIRCV	Disables the force-writing of before-images for before-image recovery and turns off after-imaging.
–NO_LOCK	Unlocks a file. Only the ROAM Administrator who locked a file can unlock it.
–NO_TRANS_ROLLBACK	Disables before-image recovery and transactional concurrency control.
–TRANS_ROLLBACK	Enables before-image recovery and transactional concurrency control.
–WRITE_ACCESS	Permits runtime updates to an activated file that has not been rolled forward.

For further information on SET_RBF and ROAM files, see the *ROAM Administrator's Guide*.

▶ **SET_SEARCH_RULES** [{ **pathname [options]** / **–DEFAULT listname** }]

SET_SEARCH_RULES initializes the user's search lists.

SET_SEARCH_RULES is used to set a search list to values specified by the user, or to reset it to system defaults. PRIMOS automatically sets special-purpose search lists to system defaults when a process is initialized. You can use SET_SEARCH_RULES either to modify these system defaults or to set a search list for which there are no system defaults. You can use the LIST_SEARCH_RULES command, described earlier in this chapter, to display the contents of a search list, whether it is your own list or a system default list.

The system typically provides the following five search lists with default values: ATTACH$, BINARY$, COMMAND$, ENTRY$, and INCLUDE$ (only ENTRY$ is required). Default values for these search lists are located in the directory SEARCH_RULES*. PRIMOS automatically sets these search lists to system defaults when you initialize your user process by logging in or issuing an ICE command.

Creating Your Own Search Rules File

A search rules file is an ASCII file of search rules that you can create with a text editor. A search rule is either a pathname or a search rule keyword. The available general-purpose keywords are –INSERT, –OPTIONAL, –SYSTEM, [**home_dir**], [**origin_dir**], and [**referencing_dir**]. Additional special-purpose keywords are provided for specific search lists. Search rule keywords are further described in the *Advanced Programmer's Guide, Volume II: File System.*

To create a search rules file, use a text editor, such as ED or EMACS, and create a file in your directory. It is recommended that the name of this file include the suffix **listname**.SR. Into this file write one search rule per line. Pathnames should be appropriate to the type of search list: for ATTACH$ you specify disk partition names; for BINARY$, COMMAND$, and INCLUDE$, you specify pathnames to your directories; for ENTRY$ you specify pathnames to EPF libraries that contain entrypoints.

You can include the –SYSTEM search rule keyword in your search rules file to indicate the location at which to include the system default rules. ATTACH$ does not use the –SYSTEM keyword; instead it uses the –ADDED_DISKS keyword.

Setting Your Search List

After you have created a search rules file, you use this file to set a corresponding search list. PRIMOS searches the search list, not the search rules file. Use the following format to set a search list:

```
SET_SEARCH_RULES pathname [-NO_SYSTEM
                           -LIST_NAME listname]
```

pathname is the name of your search rules file. If **pathname** does not have a **listname**.SR suffix, use the –LIST_NAME option. The meanings of the two options are as follows:

Option	*Description*
–LIST_NAME	Instructs PRIMOS to name the search list with **listname** rather than a name derived from **pathname**. This option allows you to specify a search list name containing a maximum of 22 characters.
–NO_SYSTEM	Instructs PRIMOS not to preface your search list with the system default rules. If you do not specify this option, PRIMOS automatically includes system default rules at the beginning of the search list.

Note

The –NO_SYSTEM option does not affect System Administrator search rules, which PRIMOS always inserts at the beginning of the search list that has been designated to contain them. Users cannot move or delete administrator rules, which are separate from those rules specified by –SYSTEM.

If you wish to include the system default rules within your search list, use the –SYSTEM search rule keyword when you create your search rules file. Afterwards (if you so desire), use the SET_SEARCH_RULES –NO_SYSTEM option when you set the search list to disable the automatic placement of the default list at the beginning of your search list.

Resetting a Search List to System Defaults

If you wish to reset a search list to the system defaults, use SET_SEARCH_RULES with the following syntax:

```
SET_SEARCH_RULES –DEFAULT listname
```

The –DEFAULT option resets the named search list. In this case, you do not specify a pathname for a search rules file.

Reinitializing the process, by issuing an ICE command, for example, resets all search lists to system defaults.

Displaying the Help Facility

To display the syntax and options of SET_SEARCH_RULES, issue the following command:

```
SET_SEARCH_RULES -HELP
```

The Help facility is also displayed if PRIMOS detects an error while parsing the command.

For more information on the Search Rules facility, see the *Advanced Programmer's Guide, Volume II: File System.* For further information on system and administrator default rules, see the *System Administrator's Guide, Volume III: System Access and Security.* For further information on the ENTRYS search list, see the *Programmer's Guide to BIND and EPFs.*

▶ SET_TIME_INFO { -TIMEZONE timezone-offset | -DLST YES [start end DST-offset] | -DLST NO } *Operator Command*

SET_TIME_INFO establishes time information for the local time zone, for example, whether daylight saving time goes into effect. This information is used by the system to calculate the Universal time. This command should be issued before SETIME at cold start, and it is recommended that SET_TIME_INFO be included in the PRIMOS.COMI file. For more information, see the *Operator's Guide to System Commands.*

▶ SET_VAR variable-name [:=] value

SET_VAR defines a global variable and places it and its value in a global variable file.

Before using SET_VAR at command level, you must have an active global variable file. (Use the DEFINE_GVAR command to create or activate a global variable file.)

variable-name, which is the name of the global variable, must begin with a period (.) and can be no longer than 32 characters. If **variable-name** is an existing global variable, its value is replaced with the new value.

value is one of the following:

- A character string short enough so that the entire command line does not exceed 160 characters. A string with blanks or special characters must be enclosed in single quotation marks. Include the single quotation marks in the character count.
- A character string consisting of the logical value TRUE or FALSE.
- An integer in the range of -(2**31) to 2**31 -1 (-2147483648 to +2147483647).

The assignment symbol := is optional.

SET_VAR Example

The following example defines a global variable and then uses the LIST_VAR command to display the results of the SET_VAR command:

```
OK, set_var .report_dir <market>davis>sales>report
OK, list_var .report_dir
.REPORT_DIR                              <market>davis>sales>report
OK,
```

Note that while the name and value of the variable were given in lowercase, PRIMOS converts the name to uppercase but retains the lowercase of the value.

Using SET_VAR in CPL Programs

Use SET_VAR in CPL programs to define either global or local variables. The command format is the same as that at the command level. The rules for **name** are the same, except that names of local variables do not begin with a period. **value** is a character string of 1024 or fewer characters in length; it must be enclosed in single quotation marks if it contains blanks or special characters. For further details, see the *CPL User's Guide*.

See also DEFINE_GVAR; DELETE_VAR; LIST_VAR.

▶ SHARE [pathname] segment-number [access-rights] *Operator Command*

SHARE incorporates files into shared segments or changes the access rights of a segment.

SHARE may be issued only from the supervisor terminal. At Rev. 21.0, you are no longer required to use the OPRPRI command in conjunction with the SHARE command.

The process of incorporating shared code into PRIMOS requires the use of both the SEG SHARE command and the PRIMOS SHARE command. The SHARE command of SEG gathers a number of SEG runfiles into a single segment runfile with an address below 4001_8. Then those R-mode runfiles below segment 4000_8 must be incorporated into PRIMOS using the PRIMOS SHARE command. For complete details, refer to the *System Administrator's Guide, Volume I: System Configuration*.

▶ SHUTDN

```
ALL [–FORCE]
pdev-1 [...pdev-n] [–FORCE]
pdev –RENAME diskname
diskname-1 [...diskname-9] [–FORCE]
  –ON nodename
```

Operator Command

SHUTDN shuts down either an entire PRIMOS system or local or remote disk partitions. For complete information, see the *Operator's Guide to System Commands.*

▶ SIZE

```
pathname [–NORM]
–HELP
```

SIZE lists the size of a file system object.

To use SIZE in systems that use Access Control Lists (ACL), you must have Read (R) access to files or segment directories and List (L) access to directories. You must also have Use (U) and List (L) access to all higher directories in the tree that contains the object.

pathname is the name of the file, directory, segment directory, or access category whose size you want displayed. If the object is in your current directory, use the object's entryname. **pathname** can be a wildcard name.

The –NORM option displays the size of files in 440-halfword records instead of the default 1024-halfword records. (1 halfword equals 2 bytes.) This option has no effect on the display if **pathname** is a directory, segment directory, or access category.

The –HELP option displays the SIZE command's syntax and options. The –HELP option's usage list is also displayed if PRIMOS detects an error while parsing the command.

The information returned by SIZE depends upon the type of object specified by the pathname, as follows:

Object	*Information*
File	Number of records (each record contains 1024 halfwords or 2048 bytes), type of file, name of file, and number of halfwords. If you specify the –NORM option, records are given in 440-halfword (880-byte) size. The size of a CAM file is displayed in both logical and physical records. The logical size is the number of records that contain data, and the physical size is the number of records allocated to the file.

Directory	Number of top-level entries (files, segment directories, sub-directories, and access categories), type of directory ("pwd" for password or "acl" for access control list), name of directory, and size in words.
Segment Directory	Number of top-level entries, type of directory file ("sam SEGDIR" or "dam SEGDIR"), name of directory, and total number of entries it can hold. To obtain the size in halfwords of the directory, multiply the last number by 2.
Access Category	Number of entries and category name.

Note

Because size is logically an attribute of an object, reading an object's size with SIZE updates the parent directory's DTA, not the object's own DTA. This is also the case when you read the object's other attributes. However, using SIZE from a pre-Rev. 20.0 system on an object in a Rev. 20.0 or later directory updates the DTA of both the parent directory and the object.

When SIZE needs to print a number greater than 32767 before the word "entries" or "records", it allows 12 full character positions for the number instead of the normal 6 character positions.

SIZE Examples

The following examples illustrate the use and output of SIZE with non-wildcard names. See the following section for using SIZE with wildcard names.

Example 1: The example below illustrates using SIZE on a file.

```
OK, size star.file
    51 records in sam file    "STAR.FILE" (51550 words)
OK,
```

Example 2: The example below illustrates using SIZE with the –NORM option on the same file as in Example 1.

```
OK, size star.file -norm
   118 records in sam file    "STAR.FILE" (51550 words)
OK,
```

Example 3: The example below illustrates using SIZE on a directory.

```
OK, size <solar>planet
   128 entries in acl UFD     "<SOLAR>PLANET" (2036 words)
OK,
```

Example 4: The example below illustrates using SIZE on a segment directory.

```
OK, size prog3.seg
     7 entries in sam SEGDIR "PROG3.SEG" (65 total)
OK,
```

Example 5: The example below illustrates using SIZE on an access category.

```
OK, size gate.acat
     3 entries in access cat "GATE.ACAT"
OK,
```

Using Wildcard Names With Size

To obtain the size of all the objects in your current directory, specify a double at sign (@@) as the wildcard name, as in the following example.

```
OK, SIZE @@
     1 record  in sam file   "LOGIN.CPL" (51 words)
     1 record  in sam file   "LOGIN.ABBREV" (647 words)
     3 entries in acl UFD    "BOOK" (70 words)
     6 records in dam file   "BB.LIST" (5555 words)
     5 entries in sam SEGDIR "AREA.SEG" (65 total)
     2 entries in access cat "GUARD.ACAT"
     0 entries in pwd UFD    "PAS_DIR" (23 words)
OK,
```

To obtain the size of all the objects in or below your current directory, use the wildcard –WALK_FROM option, as in the following example:

```
OK, SIZE *>@@>@@ -WALK_FROM 1
     1 record  in sam file   "LOGIN.CPL" (51 words)
     1 record  in sam file   "LOGIN.ABBREV" (647 words)
     3 entries in acl UFD    "BOOK" (70 words)
     6 records in dam file   "BB.LIST" (5555 words)
     5 entries in sam SEGDIR "AREA.SEG" (65 total)
     2 entries in access cat "GUARD.ACAT"
     0 entries in pwd UFD    "PAS_DIR" (23 words)
     1 record  in sam file   "*>BOOK>CHAP_1" (545 words)
     1 record  in dam file   "*>BOOK>CHAP_2" (629 words)
     1 record  in sam file   "*>BOOK>INDEX" (112 words)
OK,
```

Because the PRIMOS command processor does not treewalk segment directories, the wildcard name and option in the previous example cannot produce the sizes of files in

the segment directory AREA.SEG. To obtain the size of segment directory files, use the FUTIL utility or the pathname format *>SEGMENT_DIRECTORY_NAME>@@, as in the following example:

```
OK, SIZE *>AREA.SEG>@@
     1 record in sam file "*>AREA.SEG>0" (971 words)
     1 record in dam file "*>AREA.SEG>1" (0 words)
     2 records in sam file "*>AREA.SEG>2" (2048 words)
     2 records in sam file "*>AREA.SEG>34" (2048words)
     2 records in sam file "*>AREA.SEG>35" (2048 words)
OK,
```

Note

SIZE cannot run under DOS. Use FUTIL to find the size of a file under DOS.

▶ SLIST pathname

SLIST displays the contents of a file at the user's terminal.

pathname specifies the name of the file. Use a simple filename if the file is in your current directory.

The SLIST display scrolls continuously. If TERM –XOFF is enabled, use CONTROL-S to freeze the SLIST display. Use CONTROL-Q to continue the display or CONTROL-P to abort the command. On PST 100 terminals, you can also press PAUSE to halt the SLIST display and then press PAUSE again to continue the display.

▶ SNADSC [station-name] [–LOGON logon-command] [–TERMINAL_TYPE type]

SNADSC invokes the PRIME/SNA Interactive Terminal Emulation program, which makes a supported Prime terminal emulate an IBM 3278 Display Station.

station-name specifies a device name or device group name included in the PRIME/SNA Interactive configuration. The PRIME/SNA Administrator usually supplies this name to qualified users. If you invoke SNADSC without the station name, the SNADSC program prompts you for it.

The –LOGON option specifies an IBM logon command for a particular application on the IBM host. If the IBM logon contains embedded spaces, it must be enclosed in single quotation marks.

The –TERMINAL_TYPE option specifies the type of terminal from which you are invoking SNADSC. Currently, the only supported terminal type is PT200, which is also the default.

For further information, see the *PRIME/SNA Interactive Terminal User's Guide.*

▶ SNA_ 3270

Operator Command

SNA_3270 starts, stops, displays status information on, and controls other functions of the PRIME/SNA Interactive subsystem. For detailed information on the operation and options of SNA_3270, see the *PRIME/SNA Operator's Guide*.

▶ SNA_ 3270_CONFIG [config-pathname] [options]

Operator Command

SNA_3270_CONFIG invokes the PRIME/SNA Interactive Configurator program, which allows the PRIME/SNA Administrator to create, edit, or list an Interactive configuration. For detailed information on the operation and options of SNA_3270_CONFIG, see the *PRIME/SNA Administrator's Guide*.

▶ SNA_PRINT [–NO_LOG]

Operator Command

SNA_PRINT invokes the PRIME/SNA Interactive Printer Emulation facility.

The SNA_PRINT command also enables message logging unless the –NO_LOG option is used. While using SNA_PRINT, the operator controls printer emulation sessions by entering subcommands in reponse to SNA_PRINT prompts. For detailed information on SNA_PRINT and its subcommands, see the *PRIME/SNA Operator's Guide*.

▶ SNA_SERVER

Operator Command

SNA_SERVER starts, stops, displays status information, and controls other functions of the PRIME/SNA Server subsystem. For detailed information on the operation and options of SNA_SERVER, see the *PRIME/SNA Operator's Guide*.

▶ SNA_SERVER_CONFIG [config-pathname] [options]

Operator Command

SNA_SERVER_CONFIG invokes the PRIME/SNA Server Configurator program, which allows the PRIME/SNA Administrator to create, edit, or list the Server configurator. For detailed information on the operation and options of SNA_SERVER_CONFIG, see the *PRIME/SNA Administrator's Guide*.

▶ SORT [options]

SORT sorts as many as 20 input files into a single output file.

SORT performs two types of sorts: a default sort operation and an optional merge operation. Each operation is described below.

Input File Types

SORT can process four types of input files: ASCII (also called compressed), uncompressed ASCII, variable length (also called binary), and fixed length. All input files must be of the same type, but the output file may be of a different type.

ASCII and variable length are the two default types. The type used depends on the type of key you select. (Keys are defined in the section below, Sort Operation.) If you specify an integer or real key, SORT defaults to the variable length. Otherwise, the SORT file type defaults to ASCII. See the section below, Further Information on File Types, for details on the four types of files.

SORT Options

The SORT options are

Option	*Meaning*
–BRIEF	Does not display SORT prompts at the user's terminal.
–SPACE	Deletes blank lines from the output file.
–MERGE	Specifies a merge operation, which merges a maximum of 11 presorted files into one sorted output file. SORT prompts you for the number of extra files to be merged and then prompts for their names, one per line.
–TAG	Specifies a TAG sort. (Default) Recommended for large or unordered files. A TAG sort stores input records separate from the key data. After all keys have been sorted and merged, the corresponding records are then located and output. This option cannot be specified if –NONTAG is used.
–NONTAG	Specifies a NONTAG sort. Recommended for smaller or well-ordered files. A NONTAG sort stores each input record with its sort key in the work file, which eliminates the search for records after merging but requires more disk space.

Sort Operation

The sort operation sorts a maximum of 20 unsorted input files into one sorted file. The input files can be sorted in ascending or descending (reverse) order on a maximum of 64 keys. SORT performs a stable sort; that is, it preserves the order of input for records with equal keys.

The sort operation works by comparing keys. A **key** is a field in a line of an input file. (A key can also be thought of as a portion of the line.) The data in the key is

compared against the data in the same key of all the other lines in the input file. If more than one input file exists, the key is also compared against the same keys of all the input files.

For example, if you want to rearrange the lines in an ASCII file in alphabetical order according to the first word of the line, then the key for that sort might be the first 10 characters of each line. During the sort operation, the first 10 characters of each line would be compared to the first 10 characters of the other lines. At the end of the operation, SORT would write in alphabetical order each entire line (not just the key) into the output file.

Keys

Each key has a length and a type. The length begins at a certain column of the line and continues to a subsequent column of the same line. For example, the key might start at column 1 of a line and end at column 10. Or, if you wanted to sort on a middle part of a line, the key might start at column 40 and end at column 60.

The type of a key tells SORT what type of data (alphanumeric or numeric) is in the key and how the data is stored. ASCII files (both compressed and uncompressed) can be sorted on the following seven types of keys: A or AU for alphanumeric data, and LE, LS, TE, TS, or U for numeric data (default is A). Variable-length and fixed-length files can use any key type. Table 2-4 explains each key type in detail.

You can sort on as many as 64 keys for a line. (The default number of keys is 1.) If you are sorting on only one key that begins at column 1 and ends at the last possible column of a line (for example, 80), then the line is compared as a whole.

SORT Prompts

During a sort operation, SORT prompts you for information at two places. The first SORT prompt is

```
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME - - OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
```

At this prompt, enter the name of the input file (**INPUT TREE NAME**), the name of the output file (**OUTPUT TREE NAME**), and the number of keys for the sort (**NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS**). If you are sorting on one key, you need not enter 1 because that is the default.

Enter the information on the line at which the cursor is positioned. If you are sorting only one input file, supply the two filenames as a simple parameter list. The following parameter list specifies two sort keys, with ALPHA.IN as the input file and BETA.OUT as the output file:

```
ALPHA.IN BETA.OUT 2
```

For two or more input files, use the –INPUTFILE, –OUTPUTFILE, and –KEYS keywords, as in the following example:

```
-INPUTFILE AL.IN -INPUTFILE GA.IN -OUTPUTFILE BE.OUT -KEYS 2
```

To supply other information about the files, use the other file keywords listed in the section below, File Keywords.

The second SORT prompt is

```
INPUT PAIRS OF STARTING AND ENDING COLUMNS
ONE PAIR PER LINE - - SEPARATED BY A SPACE.
FOR REVERSE SORTING ENTER "R" AFTER DESIRED
ENDING COLUMN - - SEPARATED BY A SPACE.
FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
AT THE END OF THE LINE - - SEPARATED BY A SPACE.
   "A"  - ASCII
   "I"  - SINGLE PRECISION INTEGER
   "F"  - SINGLE PRECISION REAL
   "D"  - DOUBLE PRECISION REAL
   "J"  - DOUBLE PRECISION INTEGER
   "U"  - NUMERIC ASCII,UNSIGNED
   "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
   "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
   "LE" - NUMERIC ASCII, LEADING EMBEDDED SIGN
   "TE" - NUMERIC ASCII, TRAILING EMBEDDED SIGN
   "PD" - PACKED DECIMAL
   "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
   "UI" - UNSIGNED INTEGER
DEFAULT IS ASCII.
```

At this prompt, you must enter the starting and ending columns of each key (PAIRS OF STARTING AND ENDING COLUMNS). Optionally, you can specify a reverse sort on that key (FOR REVERSE SORTING ENTER 'R' AFTER DESIRED ENDING COLUMN) and the data type of the key (FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE AT THE END OF THE LINE).

Enter R for a reverse sort (ascending sort is the default and need not be specified). The default for the data type is "A" for ASCII. Enter the information for each key on a separate line with either a parameter list or with the Key keywords listed in the section below, Using Keywords to Enter Information.

Caution

Specifying identical names for input and output files is not recommended because the disk space used by the file becomes free (and hence vulnerable) during the sort. If the space is taken over by another user during this time, a "disk full" error — and loss of the file being sorted — may result.

After the sort is completed, SORT displays the number of passes needed for the sort and the number of items (lines) placed in the output file:

```
BEGINNING SORT



 PASSES        2        ITEMS        520

[SORT-REV20.2]
OK,
```

Sort Example

The following example of a sort operation uses ALPHA.IN as the input file and BETA.OUT as the output file. The file is sorted on two keys. The first key, which contains alphanumeric data, is columns 1 to 20 of each line. The second key, which is columns 30 to 40 of the line, contains numeric data and will be sorted in reverse order.

Parameter lists are used to enter the information:

```
OK, sort
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME - - OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
alpha.in beta.out 2
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE - - SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN - - SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE - - SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII, LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII, TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
     "UI" - UNSIGNED INTEGER
  DEFAULT IS ASCII.
1 20
30 40 r ts
```

```
BEGINNING SORT


 PASSES        2        ITEMS        520

[SORT-REV20.2]
OK,
```

After the sort is completed, BETA.OUT contains the same lines as ALPHA.IN, except that they are in alphabetical order. If two or more lines have the same data in the first key, they are entered in reverse order according to the data in the second key.

The –BRIEF Option

The –BRIEF option suppresses the SORT prompts. You still have to enter the same information as in the prompted sort, using either parameter lists or keywords.

The following example illustrates the –BRIEF option, using the same user input as in the previous example:

```
OK, sort -brief
alpha.in beta.out 2
1 20
30 40 r ts

BEGINNING SORT


 PASSES        2        ITEMS        520

[SORT-REV20.2]
OK,
```

The –BRIEF option can also be used with the –MERGE option described below. The –BRIEF option is recommended for users who are familiar with the operation of SORT.

Merge Operation

A merge operation merges as many as 10 presorted files into one output file. To specify a merge operation, use the –MERGE option, as follows:

SORT –MERGE [options]

options can be one or more of the SORT options.

Merge Prompts

A merge operation has either three or four prompts, depending on what number you enter at the third prompt. The first two prompts are the same prompts as in the sort operation. Respond to these prompts in the same way as you would for a sort operation, using either parameter lists or keywords.

The third merge prompt is

```
INPUT THE NUMBER OF ADDITIONAL FILES TO BE MERGED. (MAX= 10):
```

The maximum number of files after MAX is 10 and depends on how many input files you specified at the first prompt. At the third prompt, enter the number of additional files you want to merge. If you enter a number equal to or less than the MAX number, the fourth prompt is displayed. If you enter a zero, the merge operation begins.

The fourth merge prompt is

```
INPUT FILES TO BE MERGED, ONLY ONE PER LINE.
```

Enter the name of each additional input file on a separate line.

After the merge is completed, SORT displays the number of passes needed for the merge and the number of items (lines) placed in the output file.

Merge Example

The following example illustrates a merge of three input files (ALPHA, BRAVO, and CHARLIE) into one sorted output file named DELTA:

```
OK, sort -merge
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME - - OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
alpha delta 2
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE - - SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN - - SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE - - SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII, LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII, TRAILING EMBEDDED SIGN
```

```
      "PD" - PACKED DECIMAL
      "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
      "UI" - UNSIGNED INTEGER
    DEFAULT IS ASCII.
8 10 r
1 5
INPUT THE NUMBER OF ADDITIONAL FILES TO BE MERGED. (MAX=  10): 2
  INPUT FILES TO BE MERGED, ONLY ONE PER LINE.
bravo
charlie

BEGINNING MERGE



 PASSES        2        ITEMS        950

[SORT-REV20.2]
OK,
```

Use the –BRIEF option to suppress all merge prompts.

Using Keywords to Enter Information

The examples above used simple parameter lists to enter file and key information. For more complicated sort operations, however, you use keywords rather than parameter lists to enter information.

There are two types of keywords: file keywords enter information at the first SORT prompt and key keywords enter information at the second prompt.

File Keywords

At the first SORT prompt, use the keywords listed below to specify multiple input filenames and types or to give information about record length. The keywords must be on a single line, but can be in any order.

File Keyword	*Usage*
–INPUTFILE **name**	Specifies a file to be sorted. The name can be a pathname of no more than 80 characters. Repeat this keyword for each input file.
–OUTPUTFILE **name**	Creates a file to hold the sorted output. Only one output file per sort is allowed.
–KEYS **n**	Specifies the number of keys for the sort. **n** is from 1 to 64, inclusive.

–INTYPE { COMPRESSED | UNCOMPRESSED | FIXED | VARIABLE }	Specifies the type of file to be sorted. All input files must be of the same type. If you do not give this keyword, a default file type is taken from the key type.
–OUTTYPE **type**	Specifies the file type for the output file. **type** is one of the four types of the –INTYPE keyword. If you do not specify –OUTTYPE, the output file is the same type as the input file(s).
–INLENGTH **n**	Specifies the maximum length of the input records (in bytes). The maximum value of **n** is 32760, which is also the default. You must give this keyword for fixed-length records.
–OUTLENGTH **n**	Specifies the maximum length for records in the output file. Default is the length of input records. If you specify a fixed-length record output file, you must also specify its record length.

Key Keywords

At the second SORT prompt, use the keywords listed below to specify information about the keys. The keywords must be on a single line, but can be in any order.

Key Keyword	*Usage*
–START **n**	Uses **n** to specify the first column of the key.
–END **n**	Uses **n** to specify the last column of the key.
–DESCENDING	Requests a sort in descending (reverse) order.
–TYPE **code**	Specifies a key type for the sort. **code** is a key code from Table 2-4.
–EBCDIC	Requests that the EBCDIC collating sequence, rather than the ASCII sequence, be used for sorting. Use only with A or AU key types.

Example of Using Keywords

The following example uses file and key keywords to specify information for a sort operation:

```
OK, sort -brief
-input a.one -input b.two -output c.out -keys 2
-start 1 -end 15 -descending
-start 16 -end 25 -type le

BEGINNING SORT


 PASSES        2        ITEMS        80

[SORT-REV20.2]
OK,
```

Further Information on File Types

SORT can process the four types of files listed below. The file types are defined by the records they contain. SORT has two default file types: compressed ASCII and variable length (also called binary). The type used depends on the type of key you select. SORT defaults to the variable length on integer or real keys; otherwise, the file type defaults to ASCII.

File Type	*Definition*
COMPRESSED SOURCE (ASCII)	Blank compressed record delimited by a NEWLINE character (^212). Source lines cannot contain data that may be interpreted as a blank compression indicator (^221) or a NEWLINE.
UNCOMPRESSED SOURCE (ASCII)	Uncompressed record delimited by a NEWLINE character (^212). Source lines cannot contain data that may be interpreted as a NEWLINE. These files can also contain packed decimal data.
VARIABLE LENGTH	Record stored with length (in words) stored in first word. (The first word is not included in the word count.)
FIXED LENGTH	Record containing data only, no length information. The length must be specified using the –INLENGTH or –OUTLENGTH keywords. If a NEWLINE character is appended to each record so that the file can be edited, that character must be included in the character count.

TABLE 2-4
SORT Key Types

Code	*Key Type*	*Definition*
A	**ASCII (default)**	Character strings, stored one character per byte. Their length is limited only by the length of the record.
AU	**ASCII, uppercase and lowercase sort**	Storage is identical to regular ASCII. Lowercase characters are sorted as uppercase, but are put into the output file as lowercase.
D	**Double-precision real**	Length is 8 bytes; range is ± (10** -9902 through 10**9825).
F	**Single-precision real**	Length is 4 bytes; range is ± (10** -38 through 10**38).
I	**Single-precision integer (short)**	Length is 2 bytes; range is -32767 through +32767.
J	**Double-precision integer (long)**	Length is 4 bytes; range is -(2**31) through +2**31 -1.

TABLE 2-4
SORT Key Types - Continued

Code	*Key Type*	*Definition*
LE	**Numeric ASCII, leading embedded sign**	One digit per byte. Alphabetic characters may represent digits, as shown in the insert table below. The first character represents both a digit and the sign of the field (for example, L579 represents -3579).
LS	**Numeric ASCII, leading separate sign**	Numbers preceded by "+" or "-" indicate positive or negative value. (A blank space is treated as a positive sign.)
PD	**Packed decimal**	A four-bit nibble represents each digit; the number ends with a sign nibble. A negative sign is represented by hex D in the sign nibble; any other value in the sign nibble indicates a positive number. A packed field must have an odd number of digits plus the sign; since they are stored two nibbles (digit or sign) per byte, the result is a full number of bytes. Packed decimal keys may be no more than 63 digits plus the sign.
TE	**Numeric ASCII, trailing embedded sign**	Same as LE, except that the last digit carries the sign (for example, 357R represents -3579).
TS	**Numeric ASCII, trailing separate sign**	Same as LS, except that the "+" or "-" follows the number.
U	**Numeric ASCII, unsigned**	Like plain ASCII. These are stored one digit per byte, and are limited only by the length of the record.
UI	**Unsigned integer**	Length is 2 bytes; range is 0 through 65535, inclusive.

Digit	*Positive*	*Negative*
0	0 - + {	} -
1	1A	J
2	2B	K
3	3C	L
4	4D	M
5	5E	N
6	6F	O
7	7G	P
8	8H	Q
9	9I	R

▶ SPAC

See SET_PRIORITY_ACCESS.

▶ SPOOL { pathname [options] / options }

SPOOL prints files and manages files on the spool queue.

The SPOOL command allows you to perform the following functions:

- Print a file on a system printer or plotter. SPOOL print options allow you to specify how and where the file is to be printed.
- List all files or specific files on the spool queue.
- Modify the existing printing characteristics of a file on the spool queue.
- Cancel the printing of a file in the spool queue.

Printing a File

To print a file, use the following command format:

SPOOL **pathname [print_options]**

pathname is the name of the file you want printed or plotted. Use a simple filename if the file is in your current directory. At PRIMOS Revision 22.0, **pathname** must be the first argument in the spool command syntax.

print_options, which are listed below, denote other information about how the file is to be printed.

You select a suitable printer to print your file by specifying attributes that the printer must possess. **Attributes** are named properties of important features of the printer. For example, an attribute can denote the type of paper that is required or the location of the printer. You can specify all printer attributes simultaneously with one option, –ATTRIBUTE, described below. The names for these attributes are established by the System Administrator in special files called **printer environment files**. (Use the PROP –DISPLAY command to list the attributes of a particular printer.)

You should be aware of two requirements when spooling a file:

- You must specify the printer attributes. Only a printer with all the named attributes is used, even if that printer has additional attributes. SPOOL does not inform you if the attributes you specify are not valid.
- Some of the printer attributes may be specified as *mandatory* by the System Administrator. You must include all the mandatory attributes of the device so that the spool request is processed.

When you issue the SPOOL command, the file is copied onto the spool queue (unless the –NO_COPY option is used) and is printed according to its priority on the queue. Entering a file onto the spool queue is called spooling a file. There is only one queue for each system.

When a file is entered on the spool queue, it is identified by a request number that is unique across any cold start of the Spooler subsystem. The request number is reset to one (1) when the Spooler subsystem is cold started and the queue is empty. (This request number replaces the pre-Revision 21.0 PRT**nnn** identifier for the print request.) SPOOL then displays a message listing the spooled file's request number, number of records, and the pathname of the spool file, as in the following example:

```
OK, SPOOL BALANCE.SHEET
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Request 26 added to queue, 2 records :
<ACCTNG>FORMS>BALANCE.SHEET
OK,
```

The request number is necessary for canceling or modifying the spooled file. You can also use the SPOOL –LIST command to obtain the request number of a spooled file.

Although each system has only one queue, the spoolers copies of the files to be printed may be distributed across multiple partitions. You do not have to know the names of the partitions holding the data areas; the SPOOL command automatically distributes data files if necessary.

For more detailed information about the Spooler subsystem, see the *Operator's Guide to the Spooler Subsystem.*

Printing Options

The SPOOL options for printing a file are listed below. The –LIST option, which can be used without an initial pathname, is described more fully in a following section. Only one formatting option (–COBOL, –FORTRAN, –PLOT, –NPH or –NO_FORMAT) can be given on the command line. Use –HELP only by itself on the command line.

Print Option	*Meaning*
–ALIAS alias_name	Replaces the user name in the file header and in SPOOL –LIST displays with **alias_name**, which is a name of 16 or fewer characters. –ALIAS cannot be used on a pre-Rev. 21.0 system. Also, –ALIAS cannot be used with the –DISK option because –DISK places a spool request to a specified disk on a pre-Rev. 21 system.
–AS alias	Replaces **pathname** on the file header and in SPOOL –LIST displays with **alias**, which is a name of 16 or fewer characters.

–ATTRIBUTE name [name...]

Specifies one or more device attributes to be used in spooling the file. **Attributes** are named properties of certain characteristics of the printer, such as paper type and destination. Your System Administrator supplies the valid attribute names. The option can take one or more names and each can be specified more than once for multiple attributes. These options are not position-dependent.

–COBOL

Prints COBOL-format files. This option may be used when submitting a file to a pre-Rev. 21.0 system.

–COPIES n

Prints **n** copies of the file. (**n** cannot exceed 99.) When this option is used, the file's length is considered to be its actual length multiplied by the number of copies.

–DEFER [time]

Defers printing of the file. The file remains in the spool queue but is not printed until after the time you specify. Specify **time** in either 24-hour format (00:00 = midnight) or in 12-hour format with AM or PM (12:00 AM = midnight). The colon is optional. You must **not** leave a space between the digits and the AM or PM. If you issue the –DEFER option on the SPOOL command line without specifying a time, midnight (00:00) is assumed to be the print time. If the network crosses time zones, the deferred time used is that of the system holding the queue.

–DISK disk-id

Enters the file on the spool queue of the specified remote disk on a pre-Rev. 21.0 system. **disk-id** is the name of the disk (partition). The remote disk must contain a SPOOLQ directory. (Use the STATUS DISKS command to list the disks connected to your system.) At Rev. 21.0, SPOOL ignores –DISK if it references a local disk, and you cannot use –DISK with the options –ALIAS and –NO_COPY.

–FROM m –TO n

Specifies a range of pages to print. **m** and **n** are the page numbers of the first and last pages you want to print. **m** and **n** are printed inclusively; when used with FORTRAN or COBOL format they print logical pages.

The –FROM value must not be greater than the –TO value. Both –FROM and –TO work with multiple copies of files using the –COPIES n option. Note that embedded specific escape sequences from pages prior to **m** may not take effect, because the sequences are never sent to the printer.

–FTN

Prints the file using the FORTRAN output conventions in the file. Cannot be used in some other formatting mode (–COBOL, –PLOT, –NO_FORMAT, –NPH) is specified. The characters in column one of each line in the file have the following meanings:

Character	*Meaning*
1	Eject to top of page before printing.
-	Skip 3 lines.
0	Skip 2 lines.
space	Skip 1 line.
+	Overprint last line.

–HEADER [text]

Replaces the default page header. **text** is the header you want to appear. If **text** is not supplied, the first line of the file is used. Any format commands embedded in the file that change the page header still function. If **text** contains spaces, you must enclose it in single quotation marks. Use the keyword FILE in place of **text** in order to use the name of the file as the page header.

–HELP

Displays the SPOOL syntax and options.

–LIST [request] [option]

Lists all or specific files in a spool queue. See the section below, Listing Spooled Files, for an explanation of the –LIST options.

–LNUM

Each line of the printed file is prefixed with a number enclosed in parentheses and followed by a space. A line that uses more than one line of printed output is numbered only on the first printed line. Lines that overprint the previous line are not numbered.

–NO_COPY

Prevents the SPOOL command from performing its normal operation of making a copy of the file to be printed. Instead, the contents of the file are printed from that file's present location in the file system. Therefore, any changes made to the file between the time it is spooled and the time it is printed appear in the printed file. If you specify –NO_COPY, the file to be printed must be on the same node as the spool queue that is handling the spool request. You cannot use the –NO_COPY option with password directories unless the nonowner password is null. You cannot use –NO_COPY for requests submitted to a pre-Rev. 21.0 spooler. The despooler must have access rights to read the file if –NO_COPY is used.

–NO_EJECT	Diasables the form feed, which normally occurs after the file has completed printing. Because a form feed is never performed before a banner page, use this option only when the banner page has been inhibited with the –NOHEAD option.
–NO_FORMAT	Disables normal spooler format control (pagination and header generation). Used for files containing EVFU "skip-to-channel" commands. Cannot be used with other formatting options.
–NO_HEADER	Does not print header pages.
–NOP	Inhibits overstriking by the printer. This option ensures that SPOOL properly handles the FORTRAN or COBOL format plus-sign (+) control character, and trailing carriage return characters.
–NOTIFY	Notifies you when the file has completed printing.
–NPH	Enables file pagination without producing page headers and page numbers. –NPH is not allowed with other formatting modes.
–ON system_name	Adds a spool request to a Rev. 21.0 or later spool queue that resides on a different node. **system_name** is the name of any system that is networked to the current system. If you wish to add a spool request to a pre-Rev. 21.0 system on the network, use the –DISK option.
–OPEN [pathname]	Opens a data file in the spool queue directory. Data is generated by a program and directed to the data file. The file is opened on File Unit 2, unless you specify another file unit with a –TUNIT option. The file remains open for writing until you close it with the CLOSE command. After the file is closed, it is printed and then deleted from the spool queue. This option is useful for quickly printing listings from compilations.
–PLOT [n-words]	Denotes a plot file to be printed on the system plotter. **n-words** is the decimal number of words to be read and output per raster scan. (The default is 128 for a 200 raster/inch plotter.) A file spooled with this option cannot be printed on the line printer. –PLOT cannot be used with other formatting modes.
–PROC name	Specifies the PostScript procedure that allows you to print a file on a PostScript laser printer.
–SET_FONT fontname	Specifies a particular typeface for the file. **fontname** has a maximum length of 32 characters. If **fontname** is not supported, the file is printed using the default font name.

–SET_LANDSCAPE	Specifies that the file is printed lengthwise along the paper.
–SET_PAPER_BIN {n / MANUAL}	Specifies the paper bin to print the file. **n** is a number between 1 and 9, which selects the paper bin. **MANUAL** selects a manual paper feed.
–SET_PORTRAIT	Specifies that the file is printed widthwise along the paper.
–SFI	Suppresses the printing of both the pathname and the date/time modified notifications on the banner page and the trailer page of the printed file. (The pathname still appears in the SPOOL –LIST –DETAIL display, however, even if you use the –SFI option.)
–SPOOL_WHILE_OPEN	Allows a file to print while it is open for writing to disk. If used with –OPEN, then the file is held in the spool queue directory. If an existing file is spooled with –SWO, then –NO_COPY is forced. –SWO does not guarantee immediate printing of a file. Printing starts when a printer with the correct environment is available.
–TRUNCATE	Truncates all lines longer than the printer's width as defined in the environment file. (Use the PROP –DISPLAY command to list the value of this parameter.) If this option is not specified, lines longer than the printer width wrap around two or more lines so that no printed text is lost.
–TUNIT **n**	Specifies the file unit on which **pathname** is opened with the –OPEN option. **n** is a decimal number ranging from 1 through 125. (Default of **n** is 2.)
–XLATE **mapping**	Selects an alternate character set for the printer. **mapping** is a character string with a maximum of 32 characters that you use to select the mapping. **mapping** must begin with an alphabetic character; the remaining characters may be alphabetic, numeric, or any of the following three characters: **.** **$** **_**

Example of Printing a File

The following example of the SPOOL command prints a file on a printer controlled by your local system:

```
OK, SPOOL CMD.2 -ATT LINEP3 -ALIAS COLERIDGE -COPIES 3 -DEFER 1800
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Request 7 added to queue, 12 records : <SYSONE>BYRON>CMD.2
```

The command requests that three copies of the file CMD.2 (in the current directory) be printed with the user name COLERIDGE at 6 p.m. (18:00) at the printer with the attribute named LINEP3.

Note

Your System Administrator may set the attributes file so that SPOOL issues an error message if you specify an invalid attribute. If so, spooling a file with the invalid attribute BAD_ATTRIBUTE, for example, produces the following message:

```
OK, SPOOL LOGIN.CPL -ATT BAD_ATTRIBUTE
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Error from Spooler (Spool-107):
Unknown device attribute : BAD_ATTRIBUTE
```

Check with your System Administrator for further details.

Listing Spooled Files

To check the status of a spool queue, use the following command format:

SPOOL –LIST [request] [list_options]

If you specify only the –LIST keyword, the command lists only your files in your system's spool queue, as in the following example:

```
OK, SPOOL -LIST
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

System SYSONE
Request  Time    User            File                No  Size  State
-------  ----    --------------- ------------------- --  ----  -----
24 February 88
86       20:34   FESTER          UNCLE.COMO          1   1
25 February 88
87       13:17   FESTER          LOGIN.CPL           1   2
88       13:18   COLERIDGE       LOGIN.CPL           3   6     Defer
89       13:22   FESTER          LOGIN.CPL           1   2     Defer
OK,
```

The meanings of the header labels are as follows:

Label	*Meaning*
System	The name of the system on which the spool queue is located.
Request	The request number of the spooled file and the date on which the file was spooled.
Time	The time at which the file was spooled.
User	The name (user ID) of the user who spooled the file. If the –ALIAS option was used, lists the name specified by **alias_name**.
File	The filename of the spooled file. If the –AS option was used, lists the name specified by **alias**.
No	The number of copies to be spooled.

Size	The number of records in the spooled file. One record contains 1024 words and one word equals 2 bytes.
State	The status of the spooled file. If this field is empty, the file is not printing. If the word **Print** appears, the file is printing. If the word **Defer** appears, the file's printing has been deferred.

The –DETAIL Option

The –DETAIL option displays more spool queue information than is supplied by SPOOL –LIST. The –DETAIL report always shows the system the spool queue is on, the request number, the submittal date and time of the request, the user submitting the request, the number of copies requested, its size in records, and the complete pathname of the file. In addition, –DETAIL lists the following items, if specified, one per line:

Label	*Meaning*
Attributes	The attributes given both explicitly by the user and supplied as defaults. If no attributes are specified, this line does not appear.
Options	The options supplied by the user on the command line. If no options are specified, this line does not appear.
Deferred	The earliest time at which the file will be printed.

Here is an example of the SPOOL –LIST –DETAIL output:

```
OK, SPOOL -LIST -DETAIL
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

System SYSONE
Request   Time       User                    Copies  Size  State
-------   ----       ---------------------- ------- ----  -----
24 February 88
86         20:34:04  FESTER                  1       1
   File name  <USER1>FESTER>UNCLE.COMO
   Attributes  DISK, PRINT_ROOM

25 February 88
87         13:17:20  FESTER                  1       2
   File name  <USER1>FESTER>LOGIN.CPL
   Attributes  DISK, PRINT_ROOM

88         13:18:32  FESTER                  3       6
   File name  <USR1>FESTER>LOGIN.CPL
   Attributes  DISK, PRINT_ROOM
   Deferred until 18:00:00 on 25 February 88
   Options     -ALIAS COLERIDGE
OK,
```

The –BRIEF Option

The BRIEF option gives an abbreviated spool queue report, its display being the same as that of SPOOL –LIST.

```
OK, SPOOL -LIST -BRIEF
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

System SYSONE
Request  Time     User            File                No  Size  State
-------  ----     ----------      ----------          --  ----  -----
24 February 88
86       20:34    FESTER          UNCLE.COMO          1   1
25 February 88
87       13:17    FESTER          LOGIN.CPL           1   2
88       13:18    COLERIDGE       LOGIN.CPL           3   6     Defer
89       13:22    FESTER          LOGIN.CPL           1   2     Defer
OK,
```

List Options

The following list describes the options that you can use in conjunction with the –LIST option:

List Option	*Meaning*
request	Indicates that only the specified request is listed.
–ALL	Lists all queues that the System Administrator has placed in the SPOOL*>QUEUES file.
–ATT	Lists only entries with the specified attributes, and allows one or more than one attribute to be specified.
–BRIEF	Produces an abbreviated report with a single line per request. Use this option if you do not need a detailed listing.
–DETAIL	Produces a more detailed report, including the full pathname of the file being spooled, the printer attributes that are in effect, and the options being used (if any).
–DISK	Gives a queue report for a pre-Rev. 21.0 queue on a named partition.
–FULL	Adds a line to the –DETAIL report, which identifies the partition holding the spooler copy of the file.
–NO_WAIT	Suppresses the **--More--** prompt and provides continuous scrolling.
–ON	Gives a queue report for a named node.
–USER [name]	Restricts the queue report to **name** and defaults to the current user if no name is given.

Using –LIST When Spooling a File

You can use –LIST when spooling a file, as shown in the following example:

```
OK, SPOOL LOGIN.CPL -DEFER 1800 -LIST
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Request 89 added to queue, 1 records : <SYSONE>FESTER>LOGIN.CPL

System SYSONE
Request  Time    User            File                No Size  State
-------  ----    --------------- ------------------- -- ----  -----
24 February 88
86       20:34   FESTER          UNCLE.COMO          1  1

25 February 88
87       13:17   FESTER          LOGIN.CPL           1  2
88       13:18   COLERIDGE       LOGIN.CPL           3  6     Defer
89       13:22   FESTER          LOGIN.CPL           1  2     Defer
OK,
```

Modifying a Spooled File

Use the following command format to change the printing characteristics of a file in the spool queue:

SPOOL –MODIFY request [modify_options]

request is the request number of the spooled file.

options are the SPOOL printing options listed above, except –NO_COPY, –OPEN, –SPOOL_WHILE_OPEN and –TUNIT.

The options below are supported with –MODIFY:

Modify Option	*Meaning*
–NO_SWO	Cancels the –SPOOL_WHILE_OPEN process.
–NO_XLATE	Removes the –XLATE option from a spooler request. The default mapping is used.
–NO_DEFER	Removes the –DEFER attribute.
–NO_RUSH	Removes the –RUSH attribute. (Default)
–RUSH	Gives priority to a file so that it is printed before other spooled files. If more than one file in the spool queue has this –RUSH attribute, the files are printed in the order they were spooled. Only users in the .SPOOL_ADMINISTRATOR$ group or a user at the supervisor terminal can specify this option.

If you change the attributes of a spool request, you must respecify all the attributes the changed request is to have.

Results of Using –MODIFY

The new attributes specified with –MODIFY are compared to the existing attributes of the spooled file, with the following results:

- If a new attribute conflicts with a previous attribute, the file is given the new attribute and the old one is removed.
- If a new attribute does not conflict with previously specified attributes, it is added to the file.
- If an existing attribute is not modified by a new attribute, the existing attribute is retained.
- If an attribute is modified with the –ATT option, all attributes that were previously specified with this option are overriden. For example, if you specified –ATT CARBON FLOOR3 when you spooled the file and subsequently use –MODIFY –ATT WHITE in an attempt to change CARBON to WHITE, the result is that both CARBON and FLOOR3 attributes are overriden by the new WHITE attributes.

Example of Modifying a Spooled File

The following example illustrates the use of the –MODIFY option:

```
OK, SPOOL -MODIFY 89 -NO_DEFER -LIST
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

System SYSONE
Request  Time  User            File                No  Size  State
-------  ----  --------------- ------------------- --  ----  -----
24 February 88
86       20:34 FESTER          UNCLE.COMO          1   1

25 February 88
87       13:17 FESTER          LOGIN.CPL           1   2
88       13:18 COLERIDGE       LOGIN.CPL           3   6     Defer
89       13:22 FESTER          LOGIN.CPL           1   2     Print
OK,
```

In this example, the spooled file LOGIN.CPL has the –DEFER option removed from the file after the file has been submitted on the spool queue.

Canceling a Spooled File

To cancel the printing of file on the spool queue, use the following command format:

SPOOL –CANCEL { request / –ALL } [{ –ON nodename / –DISK disk-id }] [options]

Use the –DISK option or the –ON option only if the file is in the spool queue of another disk (that is, if you spooled the file with either of these options). **disk-id** is either the name of the disk or the logical device number of the disk on a pre-Rev. 21.0 system. **nodename** is the name of the remote system with the spool queue that holds the print request you want canceled. Once a file spooled with the –DISK option or the –ON option has begun printing, it cannot be canceled.

request is the request number of the file you want canceled.

You can also cancel more than one spool request on a single command line. Request numbers must be separated by commas or blank spaces.

If you supply the –ALL option instead of the request number, all of your spool requests are canceled. Members of the .SPOOL_ADMINISTRATORS group and the user of the supervisor terminal can cancel any queued print request and any request on the local system that has started printing. If you are a privileged user, you cannot use the –ALL option to cancel another user's requests; you must use the request number.

If the file is in the spool queue of your local system, you can cancel the file while it is being printed. Files that are being printed on remote systems cannot be canceled after printing has begun.

–CANCEL can be used with the –LIST command format, but not with the –MODIFY or –ATT SPOOL command formats.

Example of Canceling a File

The following example cancels request number 87.

```
OK, SPOOL -CANCEL 87
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Request 87 cancelled

OK, SPOOL -LIST
[SPOOL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

System SYSONE
Request  Time    User            File                 No Size  State
-------  ----    --------------- -------------------- -- ----  -----
24 February 88
86       20:34   FESTER          UNCLE.COMO           1  1

25 February 88
88       13:18   COLERIDGE       LOGIN.CPL            3  6     Defer
89       13:22   FESTER          LOGIN.CPL            1  2     Print
OK,
```

See also CONCAT; PROP.

▶ SPY

SPY displays the information that MIDASPLUS uses.

SPY is an offline menu-driven utility for use with MIDASPLUS files. You can use SPY whether or not you are using MIDASPLUS. The SPY output displays all or part of the information that MIDASPLUS uses and updates during runtime. The information includes the following:

- A table of data record locks taken
- A display of internal values at user-specified time intervals during runtime
- Systemwide and user-configurable parameters

When you invoke the SPY program, it displays a Main menu of options and asks you what type of information you want to see. Enter the appropriate response to continue.

For more information, see the *MIDASPLUS User's Guide*.

▶ SQ

See SET_QUOTA.

▶ SSR

See SET_SEARCH_RULES.

▶ START [pc] [a] [b] [x] [keys]

START restarts a program in memory.

Use START to restart a program that was interrupted before it finished execution. The program was previously loaded into memory with a RESTOR or RESUME command. The program can be an EPF, a static-mode program, a CPL program, or one of certain internal PRIMOS commands.

The interruption to the program was caused by a CONTROL-P, a BREAK, a FORTRAN PAUSE statement, a CALL EXIT statement (static-mode programs only), or a runtime error.

CPL Programs, EPF Runfiles, and Internal Commands

You can have more than one suspended CPL program, EPF program, and/or internal PRIMOS command in memory. The START command restarts the last program or internal command that was suspended. The System Administrator sets the number of EPFs that you can interrupt and restart.

Use START without arguments to restart a suspended CPL program, EPF program, or internal PRIMOS command, as follows:

```
OK, START
```

The program restarts from the point of interruption.

The following example illustrates the START command with an EPF program that was interrupted with a CONTROL-P:

```
OK, RESUME CIRCLE  /* begin execution of program CIRCLE
Enter the radius of the circle:      /* user presses CONTROL-P

QUIT.
OK, START          /* user issues START to restart CIRCLE
8                  /* program was waiting for input; user enters 8
The area of the circle is 50.3349.
OK,                /* program ends and returns control to PRIMOS
```

To restart a suspended EPF from its beginning, use the RESUME command.

Static-mode Programs

Unlike CPL and EPF programs, you can have only one suspended static-mode program in memory at a time. If you interrupt a static-mode program and then execute another static-mode program, the second program overwrites the first because they execute in the same place in memory.

For static-mode programs, START initializes the processor's registers and keys from the command line (or from RVEC, for any values not specified in the command line).

To restart a suspended static-mode program, use the following command format:

START [pc] [a] [b] [x] [keys]

The arguments for START specify new values for the RVEC vector. You must specify the new values in octal. **pc** (program counter) is the starting address of the program. **a** is the A register, **b** is the B register, and **x** is the X register. **keys** stands for the status keys associated with INK and OTK instructions.

If you issue START without arguments, one of the following two actions results:

- If no static-mode program has been invoked or restored at this command level (as is the case after a CONTROL-P, for example), the interrupted program (whether it is static mode or dynamic mode) continues from the point of interruption.
- Otherwise, the static-mode program defined by the current RVEC is invoked.

If you issue START with arguments, the register settings specified by the arguments are applied to the RVEC contents (thus defining a new static-mode machine state), and that static-mode program is invoked.

Supplying new RVEC values is generally not recommended. The exception is specifying 1000 (octal) as the value for **pc**. START 1000 restarts the most recently invoked static-mode program. (This command does not, however, reinitialize variables in the program.) For further information on specifying RVEC values, see Appendix A of this guide.

If a static-mode program is interrupted and another static-mode program is subsequently invoked, an attempt to start the first program is refused with the following error message:

```
Attempt to proceed to overwritten program image. (listen_)
ER!
```

This error occurs because the machine state for the first static-mode program has been overwritten in the RVEC, and hence the system does not have the information needed to restart the program correctly. In addition, the second program may have overwritten the first program.

▶ START_DSM [options]

Operator Command

START_DSM initializes the Distributed System Management (DSM) facility on the system. The command line options allow you to set the terminal environment so that START_DSM functions properly. The DSM server can be invoked only from the supervisor terminal.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ START_ISCNSR

Operator Command

START_ISCNSR allows the operator to start up ISC_NETWORK_SERVER at the supervisor terminal if, after the operator issues the START_NET command, the system fails to start up the ISC Network server. This command may also be used to start up the server if an unexpected error causes it to log out. For more information, see the *Operator's Guide to Prime Networks*.

▶ START_LSR

Operator Command

START_LSR allows the operator to start up the Login Server at the supervisor terminal if the Login Server does not start up when the system is booted. For more information, see the *Operator's Guide to System Commands*.

▶ START_NET { [pathname] –NODE nodename [–TRACING_NODE] | –HELP }

Operator Command

START_NET starts PRIMENET.

The START_NET command can be issued only from the supervisor terminal or by a user with the user ID SYSTEM. **pathname** is the pathname of the network configuration file. If no pathname is given, the default PRIMENET*>PRIMENET.CONFIG pathname is used. **nodename** specifies the name of the local node. The –TRACING_NODE option enables the trace option when the MONITOR_NET program is used. The –HELP option displays the complete syntax and options of this command. For further details, see the *PRIMENET Planning and Configuration Guide*.

▶ START_NSR

START_NSR spawns network servers from the supervisor terminal or other privileged processes. For more information, see the *Programmer's Guide to Prime Networks*.

▶ START_NTS [config_pathname] [–HELP]

Operator Command

START_NTS initiates terminal activity for a Prime host on a LAN300 network.

START_NTS is part of the Network Terminal Service (NTS), which allows remote LTS terminals to establish a connection with a Prime host system, and also manages local asynchronous services. For a complete description of NTS, see the *NTS User's Guide*. For more information on START_NTS, see the *Operator's Guide to Prime Networks*.

▶ STARTUP pdev-0 [...pdev-3]

Operator Command

STARTUP is an obsolete command that initializes the configuration of disk drives by relating physical disk drive numbers to logical disk numbers. Physical device numbers for disks are given in the *System Administrator's Guide, Volume I: System Configuration*.

See also ADDISK.

▶ STATUS [argument]

STATUS displays the current status of various aspects of a system.

As an operator command, STATUS monitors system usage. For more information on STATUS as an operator command, see the *Operator's Guide to System Commands.*

The arguments to the STATUS command are listed below. You can specify only one argument.

Argument	*Meaning*
ALL	Displays information accessible through all other STATUS arguments except for PROJECTS.
COMM	Displays information about the system's communication controllers (except for the Prime Node Controller). The information for each controller includes the name (ICS1, ICS2, ICS3, AMLC, MDLC, HSSMLC, SMLC), type, device address (in octal), the number of operable asynchronous lines, and the number of operable synchronous lines. The controller types are as follows: DMT (50xx old style) or DMQ (51xx new style) for AMLC controllers, a 4-digit PROM set ID number for MDLC controllers, and no information for other controllers. If the system is part of the Local Area Network (LAN), STATUS COMM displays LAN300 information as well.
DEVICE	Displays the physical and logical device numbers of any assigned magnetic tape drives, as well as the user ID and user number of the user to whom the device is assigned.

DISKS	Displays information about disks running on your system, including remote disks if your system is on a network. The information includes the diskname, octal logical device number (ldev), physical device number (pdev) for local partitions, and system node name. At PRIMOS Revision 22.0, DISKS gives information about local robust partitions and local or remote mirrored partitions.
	
	Some commands use a decimal logical device number. Use STATUS DISKS to get the octal logical device number. To convert the octal logical device number to decimal, use the command function [OCTAL ldev].
	
	For example, to get the decimal equivalent of octal 70, enter
	
	`type [OCTAL 70]`
	
	PRIMOS returns
	
	`56`
	
	56 is the decimal equivalent of 70_8.

ME	Displays your user ID, your user number, your terminal's line number (in octal and decimal), and the devices you are using. If any other users (including phantoms) are using your user ID, the same information is also displayed about those users.

NETWORK	Displays the status of the full-duplex, ring, and public data networks. The information includes each system's node name and whether the system is connected ("Up" if the system is available, "Down" if not available). STATUS NETWORK also displays the nodes on the Local Area Network (LAN).
NTS	Displays information about the Network Terminal Service (NTS), including its status (started, not started) and, if it is started, the pathname of the NTS configuration file.
PROJECTS	Displays the user ID, project ID user number of each user on your system.
SEMAPHORES	Displays the value of user semaphores that are set on your system. A semaphore, which is a flag used for synchronizing processes, is used by cooperating user processes to control access to a single shared resource.
SYSTEM	Displays the version of PRIMOS that is currently running on your system.
UNITS	Displays information about the file units that you have open, including file unit number, file position number, mode, type, read/write concurrency lock, and pathname (treename).
USERS	Displays information about each user currently using your system, including user ID, user number, terminal line number (in octal and decimal), and devices in use.

If you issue STATUS without an argument, it displays information provided by the SYSTEM, UNITS, DISK, SEMAPHORE, NETWORK, NTS, and ME arguments, in that order.

When issued from the supervisor terminal, STATUS also displays other system information. For details on STATUS as an operator command, see the *Operator's Guide to System Commands* and the *Operator's Guide to System Monitoring*.

The following example illustrates the use of STATUS without an argument:

```
OK, STATUS

System B52 is currently running PRIMOS rev. 22.0
Copyright (c) Prime Computer, Inc. 1988

User MARTHA                                B52

File    File      Open    File
Unit    Position  Mode    Type  RWlock  Treename
   31  000000000   VMr     DAM   NR-1W  <USLAB>DSM*>SIT_TEXT_DBS>DSM_USA.TODAY



                                      Mirror
                                      Primary Secondary  State
Disk   Ldev   Pdev  System  Robust ------------------------------------
USLAB    0    3460
USUSR3   1   70460
PAGER    2  100461
SYSUSA   3          B29
```

```
USA1      4          B29
USA2      5          B29
USA3      6          B29
USA4      7          B29
USA5     10          B29

Sem. Value Users
---- ----- -----
 65 177777   1

Ring network


      Node      State
    B52         ****
    B29          Up
    B17         Down

NTS is not currently started

                                  User No    Line No
                                    (In Decimal)      Devices (AL in Decimal)
User
MARTHA                                 6        rem    <USUSR3> <USLAB> (from B52)

OK,
```

▶ STATUS_DSM [options]

Operator Command

STATUS_DSM displays details of both current and loaded configuration files on any node by reference to the configuration file on a particular node. The command line options allow you to set the terminal environment so that STATUS_DSM works properly. User access to STATUS_DSM is determined by the System Administrator.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ STOP_DSM [option_1 ... option_n]

Operator Command

STOP_DSM shuts down Distributed System Management (DSM) on the local node. The command line options allow you to set the terminal environment so that STOP_DSM functions properly. You can invoke this command only from the supervisor terminal.

DSM provides networked systems management from any convenient point on the network. For more information, see the *DSM User's Guide*.

▶ STOP_LSR

Operator Command

STOP_LSR allows the operator to shut down the Login Server before bringing the machine down. For more information, see the *Operator's Guide to System Commands*.

▶ STOP_NET [-HELP]

Operator Command

STOP_NET disables PRIMENET on the system and removes the system from the network. The -HELP option displays information on the command. For details, see the *PRIMENET Planning and Configuration Guide*.

▶ STOP_NTS [-HELP]

Operator Command

STOP_NTS shuts down the Network Terminal Service (NTS) function on the system. Only remote terminal connections are affected; local asynchronous connections and PRIMENET are unaffected.

STOP_NTS is part of the Network Terminal Service (NTS), which allows remote terminals to establish a connection with a Prime host system, and also manages local asynchronous services. For a complete description of NTS, see the *NTS User's Guide*. For more information on STOP_NTS, see the *Operator's Guide to Prime Networks*.

▶ SVCSW {0 1}

SVCSW sets the SVC switch.

The SVC switch controls the handling of SVC instructions in a virtual memory environment. Setting the SVC switch to 0 (the default setting) causes all SVC instructions to be trapped and processed by the system supervisor. The SVC switch is initialized to 0 by the LOGIN command.

If the SVC SWITCH is on (SVC 1), almost all SVC instructions cause a virtual trap, and SVC instructions are handled through the user's location 65_8. The class of SVC instructions always processed by the PRIMOS operating system, regardless of the SVCSW command, are those determined by FUNCTION code 5XX. Currently, the SVC instructions are RREC (for reading from disk), WREC (for writing to disk), TIMDAT (for obtaining the time and date from PRIMOS), and RECYCL (for passing control to the next user).

For detailed information on SVCSW and on SVC instructions, see the *Assembly Language Programmer's Guide*.

▶ SYNCSORT

SYNCSORT invokes the SyncSort/PRIME utility to sort, merge, or copy one or more files. This is a separately priced product that offers functionality not available with the PRIMOS Sort utility.

For more information, see the *SyncSort/PRIME Reference Guide*.

▶ T

See TIME.

▶ TCF { –AUTOPORT n / –HOST hostname } –TERMINAL terminalname [quit-signal]

TCF accesses a remote IBM host from a DPTX-configured 3277 terminal.

The TCF command invokes DPTX/TCF (Transparent Connect Facility) on a system where DPTX/TSF and DPTX/DSC are running. –HOST specifies a valid Virtual Buffer Emulator (VBE) station name (**hostname**) for the remote IBM host computer to which you want to connect. –AUTOPORT specifies the logical line number (**n**) of the synchronous line which is searched for an available terminal address (station address). **n** is a number ranging from 0 through 7, inclusive. –TERMINAL specifies the PRIMOS name (**terminalname**) of your 3277-type-terminal. Use an asterik (*) for **terminalname** to specify your current terminal.

quit-signal specifies the signal that you enter at your terminal when you want to break the connection with the host computer and return to PRIMOS command level. The quit signal, which can be a character string or a key, is one of the following:

Quit Signal	*Meaning*
–QUIT string	Specifies a string of eight or fewer characters as the quit signal. When you enter the quit string at your terminal, you must enter it exactly as you specified it on the TCF command line.
–PA n	Specifies that a Program Attention (PA) key is the quit signal. **n** is 1 or 3.
–PF n	Specifies that a Program Function (PF) key is the quit signal. **n** is a number ranging from 1 through 12, inclusive. The PF key specified with this option cannot perform any other function in an application program.
–TR	Specifies that TEST REQUEST key is the quit signal.

When you enter the quit string at the terminal, the program breaks the connection with the host, prints out a **TCF HALT** message, and returns to PRIMOS. To the host, it seems that the terminal was powered off. Some applications programs may halt abnormally. In this case consult your System Administrator.

For a detailed description of TCF, see the *Distributed Processing Terminal Executive Guide*.

▶ TERM options

TERM sets the characteristics of a terminal line.

Use TERM to change the characteristics of the asynchronous line of your terminal from their previous settings. The new settings remain in effect until you reset them or until you log out. The login procedure resets the line's settings (except for the duplex mode) to their default values. To reset the duplex mode to full duplex at login, use the –FULL option.

If you issue TERM without an option, it displays a list of the TERM options available.

The TERM options are listed below. You can specify more than one option if they do not conflict.

Option	*Meaning*
–BREAK {OFF / ON}	Enables or disables the break character. The break character (which is CONTROL-P or the BREAK key) can interrupt a running program or command. The default is BREAK ON.
–DISPLAY	Lists the current settings of the erase and kill characters, the duplex mode, and the XON/XOFF feature.
–ERASE **char**	Sets **char** as the erase character. **char** can be any character except certain PRIMOS reserved characters, such as the percent symbol (%), the single quotation mark ('), parentheses (()), the semicolon (;), and the left bracket ([). In general, use of reserved characters as erase characters is not recommended, but if you do use a reserved character, enclose that character (except delete/rubout) in single-quotation marks ('). The default erase character is the double-quotation mark ("), although your System Administrator may have selected another character.
–FULL	Sets the terminal line to full-duplex mode, which is the normal PRIMOS duplex mode. With full-duplex operation, all characters are echoed except a LINEFEED input character, which is ignored. A carriage-return input character is passed into the system as a LINEFEED and echoed to the user as carriage return followed by LINEFEED.
–HALF [{LF / NOLF}]	Sets the terminal line to half duplex. With LF, no characters are echoed except a carriage return, which echoes a NEWLINE; input is sent to the LINEFEED system in the same manner as with TERM –FULL. NOLF functions identically except that the carriage return does not echo as LINEFEED. LF is the default value for –HALF.
–KILL **char**	Sets **char** as the kill character. **char** can be any character except certain PRIMOS reserved characters, such as the percent symbol (%), the single quotation mark ('), parentheses (()), the semicolon (;), and the left bracket ([). In general, use of reserved characters as kill characters is not recommended, but if you do use a reserved character, enclose that character (except delete/rubout) in single-quotation marks ('). The default kill character is the question mark (?), although your System Administrator may have selected another character.

–NOXOFF Disables the XOFF/XON feature for the terminal line; this is the default setting. If –NOXOFF is in effect, you cannot suspend output from the computer to the terminal by using CONTROL-S.

–XOFF Enables the XOFF/XON feature for the terminal line; –NOXOFF is the default setting. With XOFF/XON enabled, output from the computer to the terminal is suspended by using CONTROL-S (XOFF); output is resumed by using CONTROL-Q (XON).

Note

At Rev. 21.0, the TERM command options –BREAK, –DISPLAY, –ERASE, –FULL, –HALF, –KILL, –NOXOFF, and –XOFF support Network Terminal Service (NTS) lines. For more inforation on NTS, see the *NTS User's Guide.*

The following example illustrates the TERM command:

```
OK, TERM -DISPLAY

Erase = "    = '242 octal
Kill  = ?    = '277 octal
Full Duplex
Xon/Xoff disabled
OK, TERM -ERASE $ -KILL * -XOFF -BREAK ON
OK, TERM -DISPLAY

Erase = $    = '244 octal
Kill  = *    = '252 octal
Full Duplex
Xon/Xoff  enabled
OK,
```

▶ THEMIS

THEMIS invokes the THEMIS logic simulation program.

THEMIS, a separately priced product, is Prime's logic simulation program that allows you to interactively examine digital circuit models. For detailed information, see the *THEMIS User's Guide* or the *THEMIS Reference Guide.*

▶ TIME

TIME displays the amount of computer time used during a terminal session.

TIME lists the current values stored in the PRIMOS time accounting registers. The three values displayed correspond to the three values in the logout message, which are as follows: All times include PRIMOS supervisor services, such as the time spent executing supervisor subroutines on the user's behalf. Some supervisor service associated with the PRIMOS scheduler is charged to the supervisor (at the

supervisor's terminal) and not the user. When the system is idle, CPU time is charged to the supervisor. Computer time does not include I/O time for diskettes or for disks connected to a type 4000 controller.

Value	*Units Given in*	*Meaning*
Connect time	Hours, minutes	Time since login.
Compute time	Minutes, seconds	Time accumulated executing commands or using programs (does not include disk I/O time).
Disk I/O time	Minutes, seconds	Time accumulated for disk input/output. Includes user-requested I/O to files and paging I/O time generated on the user's behalf.

The following example illustrates the TIME command:

```
OK, TIME
Time used: 00h 52m connect, 00m 27s CPU, 00m 26s I/O.
OK,
```

▶ TRAMLC

TRAMLC transmits or receives a file over an assigned asynchronous line between two Prime computer systems using transparent protocol.

TRAMLC is intended for transferring files when PRIMENET is not available. Both computers must use the same baud rate and both must issue the TRAMLC command, one to transmit and the other to receive. Either computer can issue the command first. Before issuing TRAMLC, assign the appropriate asynchronous line with the following command format:

ASSIGN ASYNC LINE **n** [–TO **m**]

n and **m** are the line numbers in decimal.

When you issue TRAMLC, it displays the following prompt:

```
FNAME
```

Reply by entering the filename or pathname of the file you wish to transmit or receive. TRAMLC then prompts

```
T/R LINE# BLOCK
```

You must enter three parameters on the same line for this prompt. The meaning and parameters of the prompt are as follows:

T/R	Transmit or Receive? Enter T to send a file or R to receive one.
LINE#	Enter the octal number of your asynchronous line.

BLOCK When TRAMLC transfers a file, it divides it into 60-halfword blocks and transmits one block at a time. If you want to monitor the progress of your transmission, TRAMLC will display a **`Block n transmitted`** message at your terminal. The number of messages you receive depends on the response you give to this prompt. If you enter 0, TRAMLC sends no messages. If you enter a positive integer (**n**), TRAMLC sends a message at every **n** blocks.

When the entire file has been transmitted or received, TRAMLC displays the following message:

```
FILE COMPLETE
```

If an error occurs during file transmission, TRAMLC displays an error message that gives the reason for the error and the block number of the failure.

Note

TRAMLC is designed for conveniently transmitting and receiving small files between two Prime computers and should not be used as a communications link.

▶ TRANSFER_LOG

Operator Command

TRANSFER_LOG is a utility that performs backups and recalls for audit trail files. The audit trail is a record of system events that the System Administrator monitors. For more information, see the *System Administrators Guide, Volume III: System Access and Security*.

▶ TRANSPORT pathname –MT n [options]

TRANSPORT copies file system objects from disk to magnetic tape for transporting to another Prime site.

The TRANSPORT command is similar to the ARCHIVE command except that TRANSPORT does not save ACLs and password protection (unless you specify the –SAVE_PROTECTION option). The TRANSPORT tape can therefore be restored at another site by any user.

pathname is the directory, segment directory, or file to be copied. (RBF files cannot be copied with this command.) **pathname** can be a wildcard name and can use the treewalking option. The –MT keyword specifies the tape drive on which the tape is mounted. (**n** is the tape drive's unit number.) Before using TRANSPORT, you must assign the tape drive with the ASSIGN command.

To display an online help facility on TRANSPORT, enter the command

```
TRANSPORT –HELP
```

For details on the options and operation of TRANSPORT, see the *Data Backup and Recovery Guide*.

See also LIST_TAPE; TRANSPORT_RELEASE; TRANSPORT_RESTORE.

▶ TRANSPORT_RELEASE –MT n [options]

TRANSPORT_RELEASE releases a TRANSPORT tape so that it can be overwritten.

Any user can release a TRANSPORT tape. The –MT keyword specifies the tape drive on which the TRANSPORT tape is mounted. (**n** is the tape drive's unit number.) Before using TRANSPORT_RELEASE, you must assign the tape drive with the ASSIGN command.

To display an online help facility on this command, enter the command

```
TRANSPORT_RELEASE -HELP
```

For details on the options and operation of TRANSPORT_RELEASE, see the *Data Backup and Recovery Guide*.

See also LIST_TAPE; TRANSPORT; TRANSPORT_RESTORE.

▶ TRANSPORT_RESTORE pathname [new-pathname] –MT n [options]

TRANSPORT_RESTORE restores file system objects to disk from a tape created by the TRANSPORT command.

A TRANSPORT tape created at one Prime site with the TRANSPORT command can be copied to disk at another Prime site with the TRANSPORT_RESTORE command. **pathname** is the name of the directory, segment directory, or file to be restored. **pathname** can be a wildcard name and can use the treewalking option. **new-pathname** allows you to rename an object as it is restored or to specify a location for the object other than your current attach point.

The –MT keyword specifies the tape drive on which the tape is mounted. (**n** is the unit number of the tape drive). Before. using TRANSPORT_RESTORE, you must assign the tape drive with the ASSIGN command.

To display an online help facility on this command, enter the command

```
TRANSPORT_RESTORE -HELP
```

For details on the options and operation of TRANSPORT_RESTORE, see the *Data Backup and Recovery Guide*.

See also LIST_TAPE; TRANSPORT; TRANSPORT_RELEASE.

▶ TYPE text

TYPE prints a line of text at the user's terminal or into a command output file.

text is the character string to be displayed. **text** can contain variable references, function calls, and embedded spaces. If the character string contains one or more sets of single quotation marks, TYPE removes at least the first two sets of quotation marks.

The following example shows the effect of the TYPE command:

```
OK, TYPE My name is John.
My name is John.
OK, TYPE 'Today is Wednesday.'
Today is Wednesday.
OK, TYPE 'My name is ''John'' and today is Wednesday.'
My name is 'John' and today is Wednesday.
OK,
```

TYPE is particularly useful with CPL programs. For further information on CPL programs, see the *CPL User's Guide*.

▶ UNASSIGN { device [–UNLOAD] | –ALIAS MTldn [–UNLOAD] | ASYNC –LINE n [–TO m] }

UNASSIGN releases exclusive control of a previously assigned line or device.

device is the device that was assigned with the ASSIGN command. The device codes are the same as those listed in the description of the ASSIGN command.

The UNASSIGN command can be entered at the supervisor terminal or at the user terminal to which **device** is currently assigned. When issued from the supervisor terminal, UNASSIGN unconditionally unassigns the peripheral assigned to any user. When issued from a user terminal, UNASSIGN unassigns only the device that was previously assigned to the user. On selected devices, this command turns off the device and clears the associated I/O buffers.

All devices assigned by a user are released when that user logs out. If a user has assigned a device and has left without logging out, this command releases that device from the supervisor terminal.

Unassigning Asynchronous Lines

UNASSIGN ASYNC allows you to release control of an asynchronous line or release control of a range of consecutive lines on one command line. To unassign an asynchronous line with the UNASSIGN command, use the following format:

UNASSIGN ASYNC –LINE n [–TO m]

n is a decimal line number. If you are unassigning more than one line, **n** is the first value in a series of line numbers.

–TO unassigns a consecutive number of lines: **n** is the first number and **m** is the last number in the series of line numbers that you are unassigning. The value of **m** must be greater than **n**.

Note

At Rev. 21.0, the UNASSIGN AMLC command is obsolete. Although its use is still supported, it is not recommended.

To unassign Network Terminal Service (NTS) lines, you must specify the PRIMOS line number in decimal notation. Also, the Network Terminal Service must be running in order for you to unassign NTS lines with the UNASSIGN command. For more information on NTS, see the *NTS User's Guide*.

Unassigning Tape Drives

Magnetic tape drives are unassigned by specifying a physical or logical device number. The logical device number must have been previously assigned to that particular drive and must be specified with the –ALIAS option. The two formats are as follows:

```
UNASSIGN MTpdn [-UNLOAD]
UNASSIGN -ALIAS MTldn [-UNLOAD]
```

pdn is the physical device number of the tape drive. **ldn** is the logical device number of the tape drive. The values of both numbers range from 0 to 7, inclusive. The –UNLOAD option rewinds and unloads the tape.

If an operator unassigns a user-dedicated tape drive, no message is displayed at that user's terminal. Should the user subsequently attempt to unassign the same device, an error message is displayed.

For further details on unassigning magnetic tape drives, see the *Data Backup and Recovery Guide*.

Unassigning Disks

For a disk to be assigned to a user as an I/O device, it must neither be assigned to another user nor be in the PRIMOS file system. The disk must also be specified by an entry in the Assignable Disks Table. (Refer to the description of the ASSIGN command.) A disk that has been assigned by a user cannot be entered as an argument in the ADDISK command.

If the disk is assigned to PRIMOS, it must be released by the SHUTDN command at the supervisor terminal. The operator can also use UNASSIGN to unassign disk partitions as part of system maintenance.

For information on UNASSIGN as an operator command, see the *Operator's Guide to System Commands*.

▶ UPCASE in-pathname out-pathname

UPCASE creates an uppercase-only file from a file containing both lowercase and uppercase characters.

UPCASE scans an input file and then creates an output file that contains the same data, except that all lowercase alphabetic characters are replaced with their uppercase equivalents. The contents of the input file are not changed in any way.

in-pathname is the input file and **out-pathname** is the output file.

In the following example of UPCASE, LOWERCASE.FILE is the name of the input file and UPPERCASE.FILE is the output file. The SLIST command displays the contents of both files.

```
OK, slist lowercase.file
This file contains both UPPERCASE LETTERS and
lowercase letters.
OK, upcase lowercase.file uppercase.file
OK, slist uppercase.file
THIS FILE CONTAINS BOTH UPPERCASE LETTERS AND
LOWERCASE LETTERS.
OK,
```

UPCASE is especially useful for preparing files for output to a device (such as a printer) that handles only uppercase alphabetic characters.

▶ USAGE

USAGE lists statistics on system usage.

The USAGE command invokes the USAGE subsystem, a system metering tool. This subsystem allows the operator to monitor performance factors of the PRIMOS operating system. Although USAGE is commonly used as an operator command, it can also be issued as a user command.

The default sampling mode for USAGE is manual sampling. Automatic sampling can be provided by specifying both the –FREQ and –TIMES options. If USAGE is

issued without options, it uses manual sampling and the –USER option as defaults. One or more of the following options can be specified:

Option	*Meaning*
–ALL	Displays metering information about the system, disks, and each user, at each sample time.
–BRIEF	Displays an abbreviated format of the processes and users using system resources.
–DISK	Displays only system and disk metering information at each sample time.
–FREQ **n**	Selects automatic sampling every **n** seconds. **n** must be an integer ranging from 1 through 32767, inclusive. (It is recommended that **n** be greater than 30.) If this option is used, the –TIMES option should also be specified. If you do not specify the –FREQ option, the default is manual sampling.
–ON **nodename**	Displays metering information for a remote system specified by **nodename**. The remote system must be configured for RFA and must be running PRIMOS Rev. 19.3 or greater.
–TIMES **n**	Specifies the total number of samples to be taken if automatic sampling is in effect. The command terminates after it prints **n** sets of data. **n** must be an integer ranging from 1 through 32767, inclusive. Sampling continues indefinitely if you specify the –FREQ option but not this option.
–USER	Displays metering information about the system and each user, at each sample time. This is the default mode of operation.

USAGE Operation

The operation of USAGE depends on whether you specify manual or automatic sampling. If you specify manual sampling, USAGE displays the following message:

```
OK, USAGE
[USAGE 21.0]
Type "START" to continue.

OK,
```

Enter START to continue the USAGE procedure. USAGE displays the metering information and returns to PRIMOS command level. To display another sampling of information, enter START again at the `OK,` prompt and USAGE displays the most recent differential values.

If you specify automatic sampling, USAGE monitors the system according to your specifications and returns to PRIMOS command level when it has finished.

Caution

If a user logs in or logs out during a sampling interval, incorrect or possibly negative meter values may result. You must therefore interpret per-user metering data carefully. Processes may accumulate CPU time without actually being logged in. These processes are displayed in the USAGE output.

USAGE Examples

The following example displays a manual sampling of system usage:

```
OK, usage
[USAGE 21.0] (C) Prime Computer, Inc., 1986.
Type "START" to continue.

OK, START

 28 Mar 87 15:47:03.64  dTIME=   17.25    CPU=      0.35    I/O=      0.00
 Up since 22 Mar 87 15:08:24 Sunday    CPUtot=  29400.91 I/Otot=   5374.04

    %CPU   %Idl1   %Idl2  %Error    %I/O   %Ovlp    IO/S    PF/S
    2.04   85.30    0.00   10.23    0.00    0.00    0.00    0.00

  %Clock    %FNT    %MPC    %PNC    %SLC   %GPPI    %DSK   %IOK    %NTS
    1.51    0.00    0.00    0.45    0.00    0.00    0.10    0.00    8.83

   %AMLC  %Async   %Sync    %ICS    Segs    Used   Pages    Used   Wired
    0.37    0.00    0.00    0.00    2304     610    2048    1165     346

  Locate   %Miss  %Found   %Same  %Share   Loc/S    LM/S
       6    0.00   66.67   33.33    0.00    0.35    0.00


 Usr UserID      Mem Wire Segs   CPUtime    dCPU   %CPU   I/Otime    dI/O   %I/O
   6 BAD          99    2   11     8.604   0.261  1.513     2.524   0.000  0.000
 166 NTS_SERVER  4     2    5   515.877   0.019  0.113    15.808   0.000  0.000
 167 TIMER_PROCESS
                 5     2    3  1800.260   0.055  0.320     2.284   0.000  0.000
 174 NETMAN      4     1    5  3913.975   0.134  0.777    24.500   0.000  0.000
 179 SYSTEM_MANAGER
                18     2   13    11.777   0.010  0.059    12.800   0.000  0.000

OK,
```

The following example illustrates a command line that specifies automatic sampling:

```
OK, USAGE -FREQ 1800 -TIMES 10
```

The command causes PRIMOS to monitor the system 10 times with an interval of 1800 seconds (30 minutes) between each sampling for a period of 5 hours (1800 seconds x 10 times / 3600 seconds-per-hour).

For detailed information on USAGE, including user, system, and disk I/O metering display definitions, see the *Operator's Guide to System Monitoring*.

▶ USERS

USERS displays the number of users currently logged in to the system.

USERS does not take arguments or options. The supervisor terminal is not included in the count nor are users who are logged in remotely to other systems (unless they are using NETLINK). The following example illustrates this command:

```
OK, USERS
Users = 54
OK,
```

▶ USRASR user-number

Operator Command

USRASR assigns a user number to the supervisor terminal so that it can act as a user terminal.

user-number is the user number to be associated with the supervisor terminal. This command works correctly only if the associated communications line is not enabled on the asynchronous controller. If the line is enabled, use the following command to disable it:

AMLC TTYNOP **line-number**

line-number is the number of the communications line.

To return the terminal to use as a supervisor terminal, issue the following command:

```
USRASR 1
```

Operators using certain later machines should use the following command procedure instead of the USRASR command:

1. Press the ESCAPE key twice.
2. When the control panel prompt appears, enter the following:

```
MO USER
```

If you are not certain of the method to use for your system, check the CPU handbook for your system. For further details on USRASR, see the *Operator's Guide to System Commands*.

▶ UX_TAPE source-pathname –MT n mode-options [options]

UX_TAPE either saves files to tape in a format that the UNIX CPIO and TAR utilities can read or it restores files from a tape created by either CPIO or TAR.

source-pathname identifies the objects you want to save or restore. You can use wildcarding and iteration.

–MT n specifies the unit number of the tape drive on which the reel is mounted. The drive must be online and assigned to you.

Mode Options

When you use UX_TAPE, you must use at least one of the mode options. The mode options are –LIST, –SAVE, and –RESTORE. You can use any of these options alone on the command line, or you can use –LIST in combination with either of the other two options.

The –LIST Option

When you use the –LIST option, you can also use either –SAVE or –RESTORE; you cannot use both. Use the following format for UX_TAPE with the –LIST option:

UX_TAPE source-pathname –MT n –LIST [{ **–SAVE** / **–RESTORE** }]

When You Use –LIST With Neither –SAVE Nor –RESTORE: –LIST lists the contents of the tape on your terminal, but does not restore the tape.

When You Use –LIST With –SAVE: –LIST lists on your terminal the PRIMOS name of each file successfully saved to tape.

When You Use –LIST With –RESTORE: –LIST lists on your terminal the PRIMOS name of each file successfully restored from tape.

The –SAVE Option

The –SAVE option causes UX_TAPE to save files to tape. Use the following format for UX_TAPE with the –SAVE option:

UX_TAPE source-pathname –MT n –SAVE [options]

Options Used Only With –SAVE: The following options are available only when –SAVE is specified.

–APPEND Appends the files to the tape. The files must be in the same format (CPIO-readable or TAR-readable) as the files already on the tape.

–BLOCKING_FACTOR n Specifies the blocking factor **n**, to use when writing. **n** is in the range 1-20, with the default value of 20. This option is an alternative to the –BLOCK_SIZE option, and is for users who are familiar with the UNIX concept of blocking factors. A blocking factor of 1 is equivalent to a block size of 512 bytes. –BLOCK_SIZE and –BLOCKING_FACTOR are mutually exclusive options; you cannot use both on the same command line.

–BLOCK_SIZE n Specifies the block size **n**, in bytes, to use when writing. The default value for **n** is 10K bytes. This option is an alternative to the –BLOCKING_FACTOR option, and is for users familiar with the PRIMOS concept of block size. A block size of 512 bytes is equivalent to a blocking factor of 1. –BLOCK_SIZE and –BLOCKING_FACTOR are mutually exclusive options; you cannot use both on the same command line.

The –RESTORE Option

The –RESTORE option causes UX_TAPE to restore files from tape. Follow this syntax line when using UX_TAPE with the –RESTORE option:

UX_TAPE source-pathname –MT n –RESTORE [options]

Option Used Only With –RESTORE: The following option is available only when –RESTORE is specified.

–NO_QUERY Suppresses queries when a restored file has the same name as a file already in the current directory.

Options Used With Either –SAVE or –RESTORE

The options below are used when either –SAVE or –RESTORE is part of the command line. The option's effect may change depending on the mode.

–CPIO With –SAVE: Writes the tape in a format that the UNIX CPIO utility can read.

With –RESTORE: Reads a tape written in CPIO format.

–LCASE With –SAVE: All PRIMOS pathnames convert to lowercase UNIX pathnames. For information on processing nonalphanumeric characters, see the *Data Backup and Recovery Guide*.

With –RESTORE: All lowercase UNIX pathnames convert to uppercase PRIMOS pathnames. Uppercase UNIX pathnames remain in uppercase with each uppercase letter preceded by a slash (/). Filenames that begin with numerals, the dash (-), or the underscore (_) characters are

preceded by a slash and an ampersand (/&). For information on processing nonalphanumeric characters, see the *Data Backup and Recovery Guide*.

Note

–LCASE is the UX_TAPE default for both –SAVE and –RESTORE.

–NO_REWIND With –SAVE or –RESTORE: Suppresses tape rewind after the save or restore ends.

–NO_TRANSLATE With –SAVE or –RESTORE: Suppresses the text translation function. –NO_TRANSLATE is used when the data is in binary format.

–POS n With –SAVE or –RESTORE: Positions the tape *n* filemarks from the beginning of the tape before the save or restore begins. This option is not valid for a drive with a 60MB cartridge tape.

–SWAP With –SAVE or –RESTORE: Reverses the order of the bytes in each word. This option provides compatability with systems on which data words have the least-significant byte first.

–TAR With –SAVE: Writes the tape in a format that the UNIX TAR utility can read. This is the default format for a save.

With –RESTORE: Reads a tape in TAR format. This is the default format for a restore.

–UPCASE With –SAVE: All PRIMOS pathnames convert to uppercase UNIX pathnames. For information on processing nonalphanumeric characters, see the *Data Backup and Recovery Guide*.

With –RESTORE: All UNIX pathnames convert to uppercase PRIMOS pathnames. For information on processing nonalphanumeric characters, see the *Data Backup and Recovery Guide*.

Caution

Do not use this option unless all filenames are unique regardless of case. For example, using –UPCASE, the UNIX files README and ReadMe both translate to the PRIMOS file README. One file overwrites the other in the translation process because only one file called README can exist in the PRIMOS directory.

For more information, see the *Data Backup and Recovery Guide*.

▶ V

See VRTSSW.

▶ VPSD

VPSD loads and starts the V-mode version of the Prime Symbolic Debugger (PSD).

When the VPSD program starts, it displays a dollar-sign prompt ($) and waits for a debugging command. For detailed information, see the *Assembly Language Programmer's Guide*.

See also DBG; HPSD; PSD; PSD20; VPSD16.

▶ VPSD 16

VPSD16 loads the V-mode version of the Prime Symbolic Debugger (PSD) at 160000_8 and starts it. The commands and operation of VPSD16 are the same as those for VPSD. For further information, see the discussion of VPSD16 in the *Assembly Language Programmer's Guide*.

See also DBG; HPSD; PSD; PSD20; VPSD.

▶ VRPG pathname [options]

VRPG loads the Prime RPG II V-mode compiler and compiles an object program from an ASCII source file named **pathname**.

pathname may have either a .VRPG or a .RPG suffix, because the VRPG compiler checks for a suffix in that order. Use the BIND or the SEG command to create a runfile from the object file. (BIND is recommended.)

The compiler invoked by the VRPG command is different from the R-mode compiler that is invoked with the RPG command.

For detailed information on the operation and options of VRPG, see the *RPGII V-Mode Compiler Reference Guide*.

▶ VRTSSW [octal-number]

VRTSSW sets the virtual sense switches.

There are 16 virtual sense switches. **octal-number** is an octal representation of them as a 16-bit halfword. (Default is 0.) The number is stored and made available to the user when a program written in PMA executes an INA 1620_8 (read sense switches) instruction. The virtual sense switches are initialized to 0 by LOGIN.

The example below illustrates the VRTSSW command:

```
OK, VRTSSW 40100
```

Note

The Skip-On-Sense-Switch series of instructions always refers to the actual sense switches, not the virtual sense switches.

▶ WORD [document_name] [options]

WORD invokes the PRIMEWORD Word Processing system. If PRIMEWORD is not installed on your system, this command is not recognized by PRIMOS.

If you do not specify a document name or any options, the PRIMEWORD Main Menu is displayed. If you specify **document_name** without any options, the document is displayed and you enter the PRIMEWORD Edit function directly, bypassing the usual menus and prompts.

For further information, see the *PRIMEWORD Reference Guide*.

▶ WP_ADMIN

WP_ADMIN allows the Office Automation System (OAS) Administrator to create and maintain the Global Keyword List relating to the Document Keyword Indexing function. For detailed information on WORD, see the *OAS System Administrator's Guide*.

▶ $$

$$ is used to store a batch job's submittal options within the command file itself.

If used, the $$ command must be the first non-comment line in the command file. The complete format of $$ is

$$ JOB { * / user-name } [–options]

options are the same as the JOB command's submittal options. For more details on the $$ command, see the JOB command earlier in this chapter.

▶ * [text]

A command line that begins with an asterisk (*) signifies a PRIMOS null command, that is, a command that performs no action. **text** is any character string. However, unlike a command line beginning with a slash and an asterisk (/*), this command line is evaluated by the command processor. Thus, it is possible for this type of line to

produce error messages that will halt a program, even though the line was not intended to execute. The following series of command lines illustrate the effect of an initial asterisk:

```
OK, * This is text following an asterisk.
OK, * [This is text within brackets.]
Not found. THIS (std$cp)
Error in variable or command function reference. (std$cp)
ER! * [
Function call contains too many left brackets. (EVAL_AF)
Error in variable or command function reference. (std$cp)
ER!
```

▶ /* [text]

A command line that begins with a slash immediately followed by an asterisk (/*) is interpreted as a comment line. **text** is any character string. Unlike a command line beginning with only an asterisk, a comment line is not evaluated and therefore cannot produce error messages.

The following series of command lines illustrate comment lines:

```
OK, /* This is text following a slash and an asterisk.
OK, /* [This is text within brackets.]
OK, /* [
OK,
```

Compare this example with the previous one illustrating lines beginning with only an asterisk to see the differences between an initial asterisk and an initial slash/asterisk.

PRIMOS® Commands Reference Guide Release Notes

RLN3108-71A

July 1989

These Release Notes, RLN3108-71A, provide information about the TALK facility at Revision 22.1 to complement the material in the *PRIMOS Commands Reference Guide*.



RLN3108-71A

Copyright © Prime Computer, Inc.
Prime Park: Natick, Massachusetts 01760

▶ TALK talk_options

TALK invokes the TALK facility, an interactive message system that allows two users to conduct a real-time conversation at their terminals.

The TALK command performs the following functions:

- Begins a TALK session with another user
- Responds to a request by another user to participate in a TALK session.

Initiating a TALK Session

To initiate an interactive terminal conversation with another user, issue the TALK command using the following format:

TALK –TO { **username** / **usernumber** } **[–ON sysname] [–TERMINAL_TYPE term_type] [–STAT]**

username is the user ID of the person you wish to call. Since more than one person may be logged in as **username**, the first to respond to your TALK request becomes the other participant in the dialogue. Others logged in as **username** cannot participate.

usernumber is the user number (as defined by the STATUS USERS command) of the person you wish to call. If you send the TALK request to **usernumber**, only that user is allowed to respond.

–ON allows you to specify the remote system of the person you are calling. **sysname** is the name of the remote system as displayed by the STATUS NET command.

–TERMINAL_TYPE specifies your terminal type (PST 100 or PT200, for example). Use this option if you do not have the .TERMINAL_TYPE$ global variable set to the proper terminal type. If you do not have .TERMINAL_TYPE$ defined and you do not use –TTP, TALK will prompt you for your terminal type.

–STAT displays certain statistics at the end of the TALK session. The TALK server keeps these statistics in a logfile in the TALK* directory; the most recent logfile and next most recent logfile are preserved in TALK*. These statistics include

- The number of characters sent and received
- The number of InterServer Communication (ISC) messages sent and received
- The number of ISC delivery failures and retries

These statistics are useful in debugging problems caused by ISC and X.25 message delivery failures. For more information on ISC messages, see *Subroutines Reference V: Event Synchronization.*

Failure of TALK Requests: Your TALK request will fail if any of the following three conditions are true:

- The user you are calling is not logged in.
- The user you are calling is not accepting messages (has used either the MESSAGE –DEFER command or the MESSAGE –REJECT command to put the receiving terminal in a DEFER or REJECT state).
- The user you are calling is already participating in another TALK session.

Calling Another User: After you have successfully issued the TALK command, your terminal screen divides into a top and bottom window. Your user ID appears in the upper left portion of the top window, and the user ID of the person you are calling (when that person answers) appears in the upper left portion of the bottom window. A reminder that the CTRL D key sequence terminates the session appears on the lower right portion of the bottom window. At this point, you cannot type any text until the called user responds to your request.

If the called user does not respond, or if you wish to terminate the session before the called user reponds, type CTRL D: this sends control of your terminal back to PRIMOS and sends a message to the called user that you have terminated the session.

Responding to a TALK Request

When another user calls you to participate in a TALK session, you will see the following message on your terminal screen:

```
***TALK_SERVER (user nnn on sys_name) at hh:mm
user_name on sys_name is calling you.  Use TALK -RESPOND to answer.
```

If you do not wish to participate in the TALK session, ignore the request. If you choose to participate in the TALK session, issue the TALK command in the following format:

```
TALK -RESPOND [-TERMINAL_TYPE term_type] [-STAT]
```

–TERMINAL_TYPE and **–STAT** have the same meanings as documented above in the section *Initiating a TALK session*. After you have issued the TALK –RESPOND command, your terminal screen also divides into a top and bottom window. Your user ID appears in the upper left portion of the top window, and the user ID of the person to whom you are responding appears in the upper left portion of the bottom window. A reminder that the CTRL D key sequence terminates the session appears on the lower right portion of the bottom window.

Note

You cannot initiate a TALK session at the supervisor terminal (User 1), nor can you respond to a TALK request at the supervisor terminal.

While Both Users Are In a TALK Session

When two users are engaged in a TALK session, both of them may enter text at any time. The text appears in the appropriate window of each terminal as the user types that text.

TALK performs word wrapping if the user enters text past the edge of the window's right margin. When the text has completely filled the window, the text scrolls out of the window one line at a time thereafter; the text that has scrolled out of the window cannot be saved.

You can use three non-text commands:

- CTRL G rings the terminal bell of the called user.
- CTRL L refreshes the terminal screen.
- CTRL D terminates the TALK session.

Terminating the TALK Session

When you wish to end the TALK session, type CTRL D. This refreshes your terminal screen, and also refreshes the screen of the other participant. Both of you will then see the following message:

```
[TALK  Rev. 22.1   Copyright (c) 1988, Prime Computer, Inc.]
Session terminated.
```

If you used the –STAT option when you issued the TALK command, you will also see the statistics from the TALK session. Here is an example:

```
[TALK  Rev. 22.1   Copyright (c) 1988, Prime Computer, Inc.]
Session terminated.

Characters sent =          176.
Characters received =      238.
Messages sent =             10.
Messages received =        103.
Delivery failures =          0.
Total retries =              0.
```

Note

Do not use CTRL P to terminate a TALK session, as this may suspend your terminal process; use CTRL D instead.

Getting Help

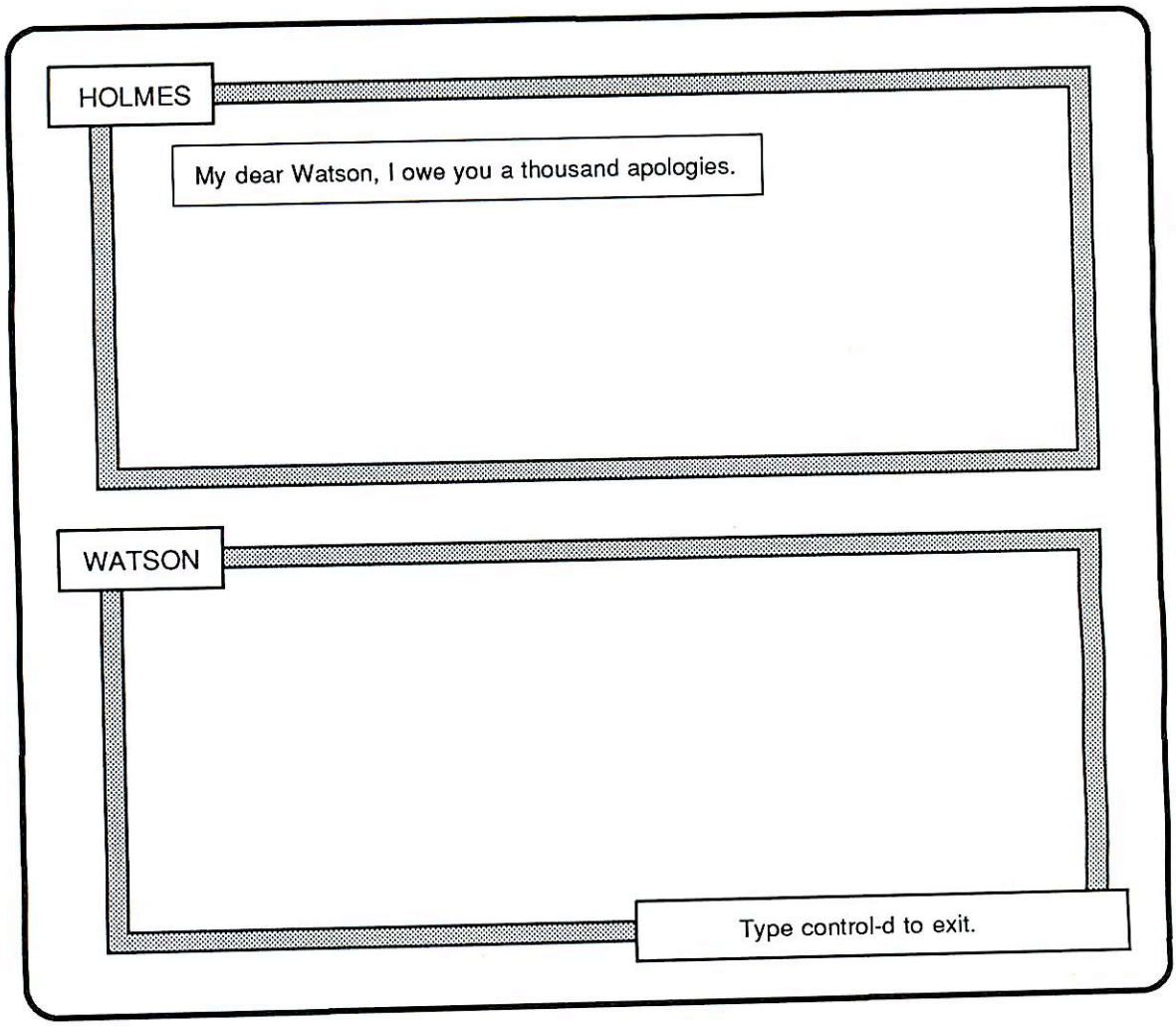
If you need assistance in using TALK, issue the following command:

```
TALK -HELP
```

-HELP displays a list of the command line options and their abbreviations. You cannot use this option with any other options on the command line.

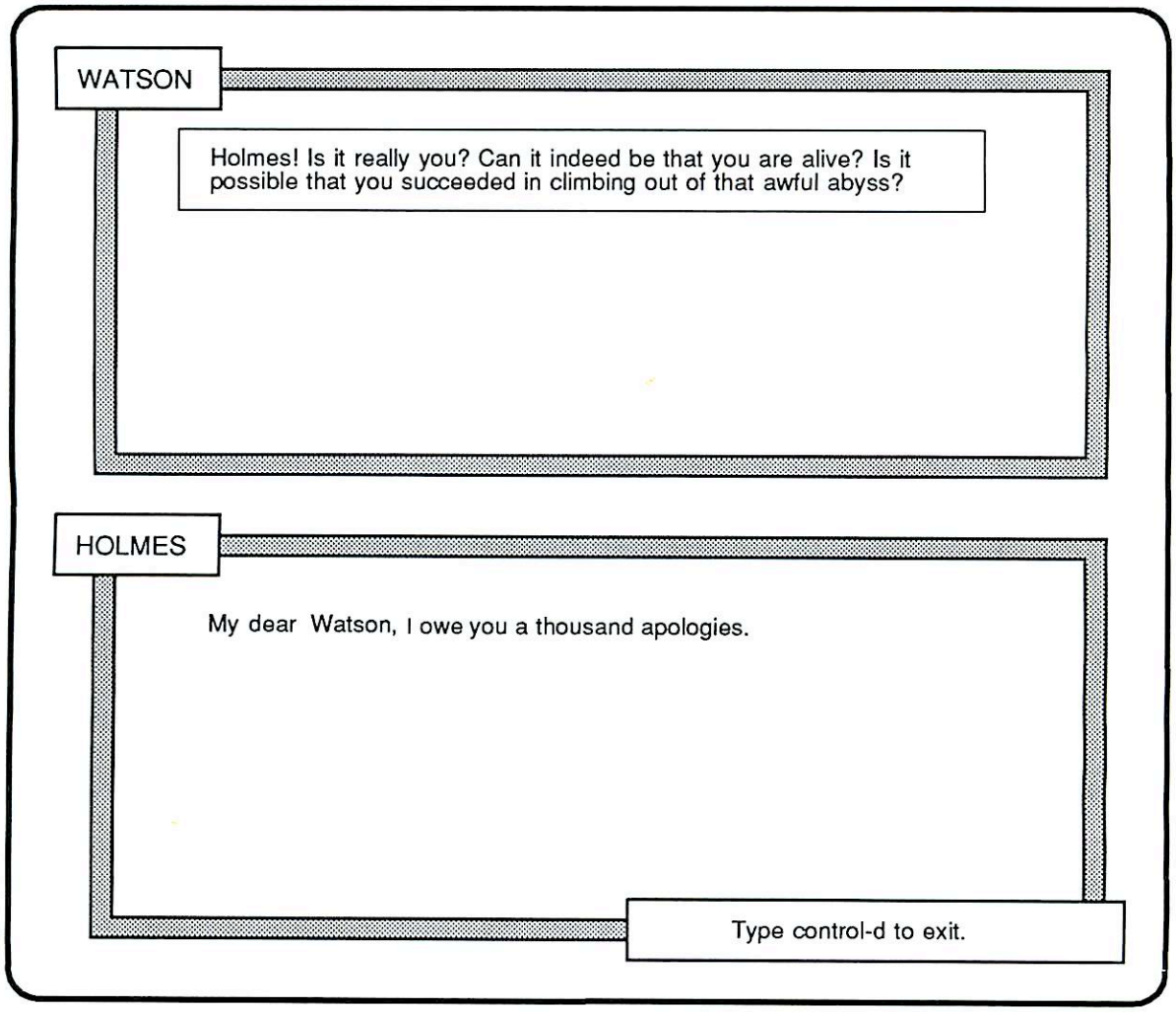
An Example Session

In the following TALK session, HOLMES has initiated the conversation and WATSON has responded. The first screen example shows HOLMES' terminal screen as he begins the conversation. Text as it is being entered is boxed.



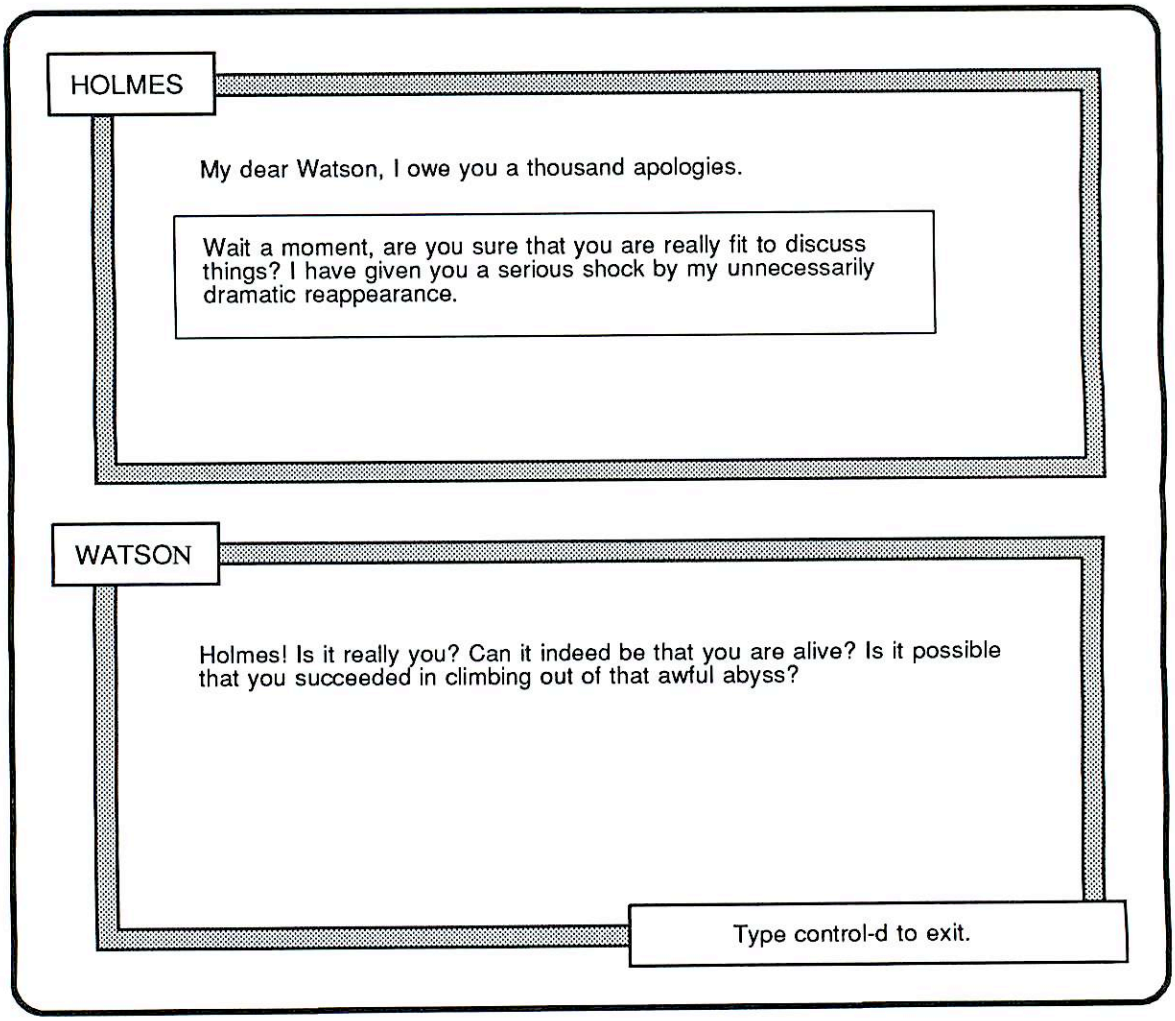
On HOLMES' Terminal: HOLMES' First Message

The second screen example shows WATSON at his terminal responding to HOLMES' first message. Notice that on WATSON's terminal, the windows are reversed, with WATSON's window being uppermost on his terminal screen.



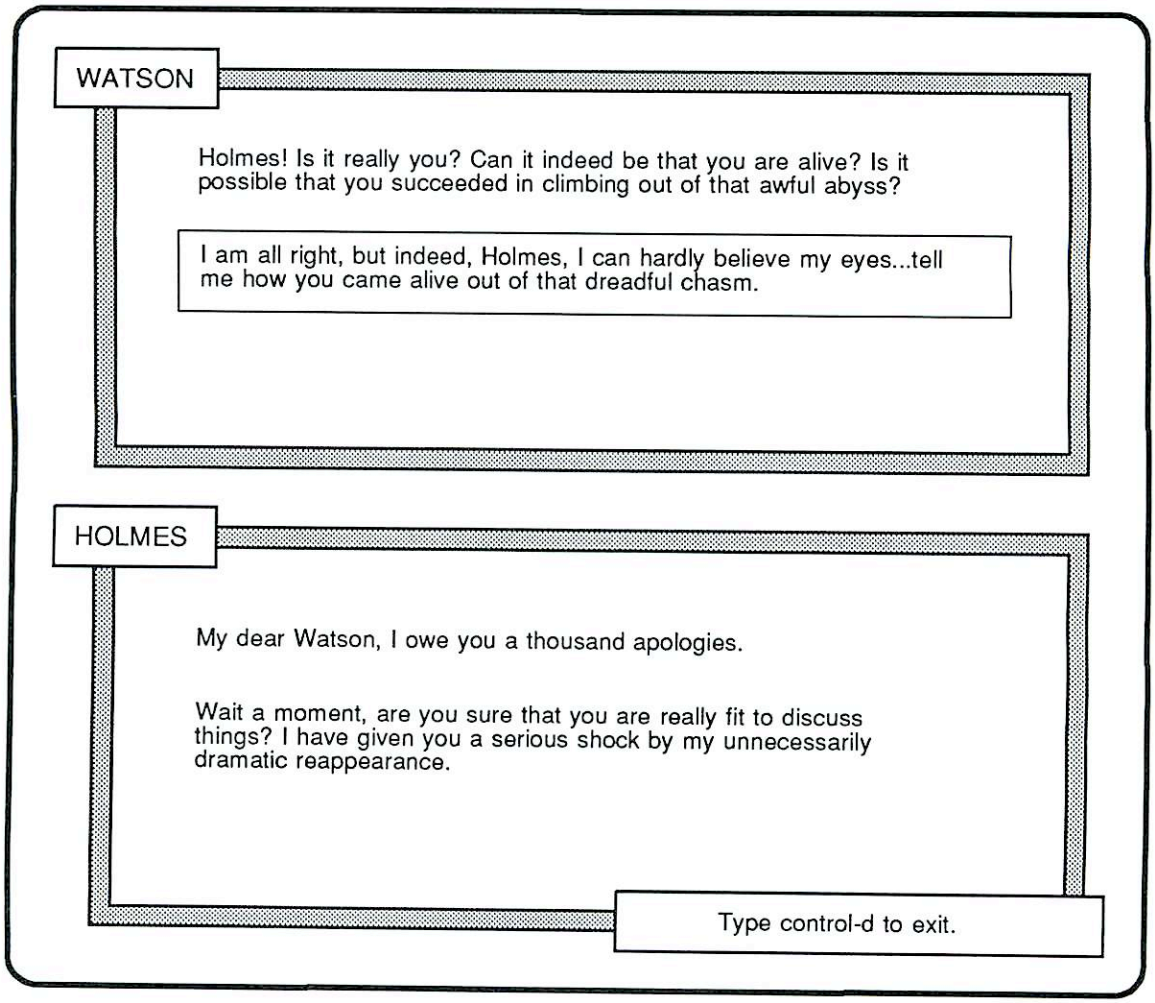
On WATSON's Terminal: Response to HOLMES' First Message

The third screen example shows HOLMES' screen as he sends a message that answers WATSON's response. HOLMES has entered several carriage returns before beginning his second message in order to provide some visual separation from his previous message. If he had not separated his own text, his second message would have started immediately after his first message at the cursor position.



On HOLMES' Terminal: HOLMES' Second Message

The fourth screen example shows WATSON's response to HOLMES' latest remarks. WATSON has also entered several carriage returns before beginning his second message in order to set off his latest text from his previous message. Text remains on the screen until the window is filled, at which time text begins scrolling out of the top of the window one line at a time.



On WATSON's Terminal: Response to HOLMES' Second Message

3 Dictionary of Command Functions

Introduction

Command functions are intended primarily for use with the Prime Command Procedure Language, CPL. However, they may be invoked at command level as well. At command level, these functions are most commonly invoked in conjunction with the TYPE command.

Functions and their arguments are always enclosed in square brackets. For example,

```
TYPE [CALC 254 * 19]
```

When the command line is processed, the function call is evaluated. The value returned by the function call then replaces the function call itself in the command line. For example, the command line shown above would evaluate to "TYPE 4826'. Users do not see this intermediate stage, but see only the following at their terminals:

```
OK, TYPE [CALC 254 * 19]
4826
OK,
```

Several of the command functions either require or allow the use of variables. If these functions are invoked at command level, all variables must be global. Chapter 4 describes the use of global variables at command level.

If a variable is required as an argument within a function call, the variable name need not be bounded by % symbols. The % symbols are required, however, if a variable value is substituted for an integer or character string argument in a function call. For example, .VAR1 requires % symbols in the following function call, which can accept either variables or integers:

```
[CALC %.VAR1% / 2]
```

However, the variable .VAR1 does not require % symbols in the following function call, which accepts only variables as its arguments:

```
[GET_VAR .VAR1]
```

Command functions fall into four categories: arithmetic functions, file system functions, string-handling functions, and miscellaneous functions. The functions are listed by category in Chapter 1.

For detailed information on user-written command functions and on calling command functions from programs, see the *Advanced Programmer's Guide, Volume III: Command Environment*.

Command Functions

▶ [ABBREV { –EXPAND text / –STATUS }]

ABBREV expands the value of an abbreviation or returns the status of an abbreviation file.

The ABBREV –EXPAND format retrieves the value of **text** from the currently active abbreviation file, and returns the expanded value as its result. If **text** is not an abbreviation, **text** itself is returned.

If, for example, the abbreviation AM stands for ATTACH DIR>MYDIR, then

```
[ABBREV -EXPAND AM]
```

returns

```
ATTACH DIR>MYDIR
```

The ABBREV –EXPAND format treats **text** as a command line and expands it. Only command abbreviations (that is, abbreviations created by the commands ABBREV –ADD and ABBREV –ADD_COMMAND) are expanded. To expand argument abbreviations, put a null string ('') in the command position, as follows:

```
[ABBREV -EXPAND '' text]
```

The ABBREV –STATUS format returns the absolute pathname of your currently active abbreviation file. For example, if the name of your abbreviation file is LOGIN.ABBREV and is in the directory <FRUIT>PEACH, then

```
[ABBREV -STATUS]
```

returns

```
<FRUIT>PEACH>LOGIN.ABBREV
```

If you turn off your abbreviation file with the ABBREV –OFF command, the ABBREV –STATUS function displays the pathname of the file followed by the word –OFF, as in the following example:

```
OK, ABBREV -OFF
OK, TYPE [ABBREV -STATUS]
<FRUIT>PEACH>LOGIN.ABBREV -OFF
OK,
```

If you have not activated your abbreviation file since you logged in, ABBREV –STATUS returns only –OFF.

Invocation of the ABBREV function fails if abbreviations are not enabled at your site.

▶ [AFTER string find-string]

AFTER prints all text or characters that appear after the specified text or characters.

string is the text or characters to be searched and **find-string** is the text or characters to be located. For example,

```
[AFTER HELLO E]
```

returns

```
LLO
```

If **find-string** is not in **string** or is at the end of **string**, AFTER returns the null string.

▶ [ATTRIB pathname {attribute-selection options} [–BRIEF]]

ATTRIB returns information about a specified file system object.

pathname is the name of the file, directory, segment directory, or access category. **attribute-selection options** are listed below. You must specify one of these options.

Option	*Meaning*
–TYPE	Returns the file system type of **pathname**: • **ACAT**, for an access category • **CAM**, for a ROAM contiguous access file • **DAM**, for a direct access file • **SAM**, for a sequential access file • **SEGDAM**, for a segmented direct access file • **SEGSAM**, for a segmented sequential access file • **UFD**, for a directory • **UNKNOWN**, if a file is not of a recognized type
–DTA	Returns the date and time an object in a Rev. 20.0 or later directory was last accessed. The date and time are in the format of DATE –FULL, that is, yy-mm-dd.hh:mm:ss.day. Objects with undefined DTA values return `** dta not set **`
–DTB	Returns the date and time the object was last backed up by the BACKUP utility. Objects with undefined DTB values return `** dtb not set **`
–DTC	Returns the date and time an object in a Rev. 20.0 or later directory was created. Objects with undefined DTC values return `** dtc not set **`
–DTM	Returns the date and time the object was modified.
–LENGTH	Returns the length of the object in two-byte halfwords.
–SIZE	Returns the same information as –LENGTH.

Note

Because size is logically an attribute of an object, reading an object's size with either –SIZE or –LENGTH updates the parent directory's DTA, not the object's own DTA. However, using either –SIZE or –LENGTH from a pre-Rev. 20.0 system on an object in a Rev. 20.0 or later directory updates the DTA of both the parent directory and the object.

If you specify –BRIEF, most error messages produced by the function are suppressed. Only error messages indicating improper invocation of the function or its arguments are printed.

▶ [BEFORE string find-string]

BEFORE prints all text or characters that appear before the specified text or characters.

string is the text or characters to be searched and **find-string** is the text or characters to be located. For example,

```
[BEFORE HELLO L]
```

returns

```
HE
```

If **find-string** is not in **string**, BEFORE returns **string**. If **find-string** is at the beginning of **string**, BEFORE returns the null string.

▶ [CALC expression]

CALC evaluates any arithmetic and logical expressions found in **expression**.

CALC accepts expressions containing the logical operators & (and), | (or), and ^ (not); the arithmetic operators +, -, *, /, unary +, and unary -; and the relational operators =, <, >, <=, >=, and ^=. Note that exponentiation is not supported. The operator precedence is as follows:

```
Highest:   ^   unary +   unary -
           /   *
           +   -
           =   ^= < > <= >=
           &

Lowest:    |
```

Note

All operators evaluated by CALC must be delimited by blanks. This restriction resolves the ambiguity that can arise from the fact that *, <, and > are also valid pathname characters. If parentheses are used within **expression**, each "(" and ")" must be bounded by blanks.

Logical and relational operators return the Boolean values TRUE and FALSE. For example,

```
[CALC 9 > 3]
```

returns

```
TRUE
```

Relational operators accept either numeric or nonnumeric operands. If a relational operator is given a nonnumeric operand, an ASCII-based string comparison is done. If both operands are numeric or Boolean, an arithmetic comparison is done. Boolean TRUE is interpreted as 1 and FALSE as 0.

Arithmetic operators must have, as operands, integers or variables that convert to integers. The values of the operands, as well as the value returned by the function call, must be in the range -2147483648...+2147483647. (This is the range of signed values that can be represented by a 32-bit integer.)

Arithmetic operators return a character string representation of the numeric result. Arithmetic operators apply only to integer values; no floating-point arithmetic is supported at command level or in CPL.

All the arithmetic operators have the usual definition, except for / which returns only the truncated integer part of any noninteger result. For example,

```
[CALC 99 / 25]
```

returns

```
3
```

▶ [CND_INFO flag]

CND_INFO allows a CPL condition handler to examine the condition information of the most recent condition on the stack.

CND_INFO returns information requested by **flag** as follows:

-NAME	Returns the name of the condition. If no condition frame is on the stack, $NONE$ is returned.
-CONTINUE_SWITCH	Returns the Boolean value of the continue-to-signal switch. If no condition frame is on the stack, FALSE is returned.
-RETURN_PERMIT	Returns the Boolean value of the returned permitted switch. If no condition frame is on the stack, FALSE is returned.

For further information on CMD_INFO, see the *CPL User's Guide*. For further information on the Prime Condition Mechanism, see the *Subroutines Reference III: Operating System*.

▶ [DATE [option]]

DATE returns the current date and/or time in a variety of formats.

If you omit **option**, only the date is returned: 87-07-14. If you use one of the options, the date and/or time is returned as follows:

Option	*Format*
–AMPM	`11:05 AM`
–CAL	`July 14, 1987`
–DAY	`14`
–DOW	`Tuesday`
–FTAG	`840714.110508`
–FULL	`84-07-14.11:05:08.Tue`
–MONTH	`July`
–TAG	`870714`
–TIME	`11:05:08`
–USA	`07/14/87`
–UFULL	`07/14/87.11:05:08.Tue`
–VFULL	`14 July 87 11:05:08 Tuesday`
–VIS	`14 July 87`
–YEAR	`1987`

▶ [DIR pathname [–BRIEF]]

DIR returns the directory portion of **pathname**, that is, all of **pathname** except its final component.

For example,

```
[DIR JOHN>DATA>EXP1]
```

returns

```
JOHN>DATA
```

* is returned if **pathname** is a simple filename.

If you specify –BRIEF, most error messages produced by the function are suppressed. Only error messages indicating improper invocation of the function or its arguments are printed.

▶ [ENTRYNAME pathname]

ENTRYNAME returns the entryname portion of **pathname**, that is, its final component.

For example,

```
[ENTRYNAME JOHN>DATA>EXP1]
```

returns

```
EXP1
```

pathname itself is returned if it is a simple filename. For example,

```
[ENTRYNAME HELLO]
```

returns

```
HELLO
```

▶ [EXISTS pathname [type] [–BRIEF]]

EXISTS is a Boolean function that determines

- Whether a file system object exists
- Whether the file system object is of a specified type (file, directory, segment directory, or access category)

pathname is the name or pathname of a file or directory.

type may be omitted or may be one of the following:

- –ANY
- –ACCESS_CATEGORY
- –FILE
- –DIRECTORY
- –SEGMENT_DIRECTORY

The value TRUE is returned if **pathname** exists and matches the file type specified. The value FALSE is returned if **pathname** cannot be found or does not match the file type specified. If **type** is –ANY or is omitted, only the existence of **pathname** is checked.

For example, assume the current directory contains a file called HELLO. The function calls

```
[EXISTS HELLO -ANY]

[EXISTS HELLO]

[EXISTS HELLO -FILE]
```

all return

```
TRUE
```

because HELLO exists and is a file. However, the function call

```
[EXISTS HELLO -DIRECTORY]
```

returns

```
FALSE
```

because HELLO is not a directory.

If you specify –BRIEF, most error messages produced by the function are suppressed. Only error messages indicating improper invocation of the function or its arguments are printed.

▶ [GET_VAR expr]

GET_VAR returns the value of the variable name defined by **expr**.

The string $UNDEFINED$ is returned if the variable named by **expr** has not been defined, or if no global variable file is active.

GET_VAR is useful for testing whether a variable has been set. If the function is used at command level, only global variables may be accessed. For example, if .VAR1 has been set to 32, but .VAR2 has not been set to any value, then

```
[GET_VAR .VAR1]
```

returns

```
32
```

but

```
[GET_VAR .VAR2]
```

returns

```
$UNDEFINED$
```

▶ [GVPATH]

GVPATH returns the pathname of your active global variable file.

If, for example, your active global variable file is <TOP>DIR>VARFILE, then

```
[GVPATH]
```

returns

```
<TOP>DIR>VARFILE
```

If no global variable file is defined or active, the function returns

```
-OFF
```

▶ [HEX number]

HEX converts a nonnegative hexadecimal number to its decimal equivalent.

number is the hexadecimal number or letter to be converted. For example,

```
[HEX C]
```

returns

```
12
```

▶ [INDEX string find-string]

INDEX locates and prints the starting position number of a substring in a string.

string is the text to be searched and **find-string** is the text or characters to be located in **string**. For example,

```
[INDEX ABCDEFGHIJ GH]
```

returns

```
7
```

▶ [LENGTH string]

LENGTH prints the number of characters in a given character string.

For example,

```
[LENGTH HOW LONG IS THIS?]
```

returns

```
17
```

Leading, trailing, and multiple blanks within **string** are not counted by the function unless **string** is quoted.

▶ [MOD number1 number2]

MOD divides one number by another and returns the remainder (modulus).

number1 is the dividend and **number2** is the divisor. For example,

```
[MOD 299 100]
```

returns

```
99
```

▶ [NULL string]

NULL tests a string for the occurrence of any text or characters and returns TRUE if no text or characters exist and FALSE otherwise.

string is the text or characters to be tested. For example, if the variable .VAR3 has been set to a null string, then

```
[NULL %.VAR3%]
```

returns

```
TRUE
```

▶ [OCTAL number]

OCTAL converts a nonnegative octal number to its decimal equivalent.

number is the octal number to be converted. For example,

```
[OCTAL 12]
```

returns

```
10
```

▶ [OPEN_FILE pathname status-var –MODE m]

OPEN_FILE opens a file for reading or writing. It returns the unit number, a decimal integer indicating the file unit on which it opened the file.

pathname is the name or pathname of the file to be opened.

status-var is the name of a global or local variable that is automatically set to 0 if the operation is successful and nonzero otherwise. If OPEN_FILE is used at command level, **status-var** must be a global variable. **status-var** may be either global or local if OPEN_FILE is invoked within a CPL program.

–MODE **m** indicates how a file is to be opened: reading only, writing only, or reading and writing. **m** must be one of the following:

- R or r, to specify reading only
- W or w, to specify writing only
- WR or wr, to specify reading and writing

It is often convenient to set a variable to the file unit number by using the SET_VAR command in conjunction with the OPEN_FILE function, as follows:

```
OK, SET_VAR .UNIT [OPEN_FILE YOYO .STAT -MODE R]
OK,
```

%.UNIT% may then be supplied as the file unit number when the READ_FILE or WRITE_FILE function is invoked. The variable must be global if it is set or used at command level. It may be either local or global if it is used within a CPL program.

▶ [PATHNAME path [–BRIEF]]

PATHNAME returns the full pathname (including the disk name) of **path**.

For PATHNAME to work, the entire directory portion of **path** must exist and the user's process must be able to attach to that directory. However, the final element in the pathname need not exist; if you supply PATHNAME with a nonexistent filename, it returns the same value that it would return if the filename exists.

If **path** is either an entryname or an asterisk (*), PATHNAME returns the full pathname of the current directory with either the entryname or * appended to it. If, for example, the current directory is ROOM, a subdirectory of HOUSE on the disk TOWN, then

```
[PATHNAME *]
```

returns

```
<TOWN>HOUSE>ROOM>*
```

If **path** is a relative pathname, then its elements are appended to the pathname of the current directory. Thus, continuing the example above,

```
[PATHNAME *>KEY]
```

returns

```
<TOWN>HOUSE>ROOM>KEY
```

If **path** is a pathname that includes one or more right-angle brackets (>) and does not begin with an asterisk (*), PATHNAME assumes that **path** is a full pathname, that is, PATHNAME assumes that the specified pathname begins with a top-level directory. If the directory <FARM>BARN exists and is accessible, then the function call

```
[PATHNAME BARN>STALL]
```

returns

```
<FARM>BARN>STALL
```

If <FARM>BARN does not exist or is not accessible, an error message is generated.

The –BRIEF option suppresses most error messages produced by the function. Only error messages indicating improper invocation of the function or its arguments are displayed.

▶ [QUERY text [default] [–TTY]]

QUERY prints the character string **text** on the terminal screen and follows it with a question mark.

If **text** is null, printing is suppressed. **text** and **default** must be quoted if they contain special characters or embedded blanks. One level of quoting is stripped before printing. After **text** has been printed, the system waits for you to enter YES, Y, OK, NO, or N. You may use either uppercase or lowercase characters.

QUERY returns TRUE if you enter YES, Y, or OK. It returns FALSE if you enter NO, N, NEXT, QUIT, or Q. The default value is returned if you enter a carriage return. If no default value is supplied, FALSE is assumed to be the default value.

The example below illustrates the QUERY function:

```
OK, TYPE [QUERY HELLO]
HELLO? YES
TRUE
OK,
```

The –TTY option forces the QUERY function to take input from the terminal. If you omit this option, the function takes its response from the command input stream. That is, if the function is invoked interactively, it goes to the terminal for its response. If the function is invoked from a command input file, or from a &DATA group within a CPL program, it goes to the command input file or CPL program for its input. A CPL program containing the function QUERY with the –TTY option cannot be executed as a batch job or phantom.

▶ [QUOTE string1 string2 ... stringn]

QUOTE places an outer pair of quotes around the text specified in **string** and doubles the quotes already inside the given string.

The QUOTE function is used to keep the meaning of special symbols from being interpreted during calls to subsystems. For example,

```
[QUOTE XY'|'Z]
```

returns

```
'XY''|''Z'
```

Note that the TYPE command removes a level of quotes from its argument, so that the following results are obtained when TYPE and QUOTE are used together:

```
OK, TYPE [QUOTE XY'|'Z]
XY'|'Z
OK,
```

▶ [READ_FILE unit status-var –BRIEF]]

READ_FILE reads the next line from a previously opened ASCII file and returns the line as its value. The line is quoted if it contains special characters. The true null string (a string of zero length, containing no characters) is returned if end-of-file is encountered.

unit is the file unit number of the file to be read. It may be the decimal integer returned by the OPEN_FILE function or a variable whose value was set using the SET_VAR command with the OPEN_FILE function. If you use a variable at command level, it must be a global variable.

status_var is a variable that is automatically set to 0 if the operation is successful and to a nonzero integer otherwise. When end-of-file is reached, **status-var** is set to 1. If you invoke the function READ_FILE at command level, **status-var** must be a global variable.

For example, if the file YOYO was opened on unit 35 and the next line in YOYO is "This is yoyo.", then the command

```
[READ_FILE 35 .STAT]
```

returns

```
This is yoyo.
```

and sets .STAT to 0. Subsequent calls to READ_FILE return successive lines from the file. After the last line of YOYO is read, the next call to READ_FILE returns the null line and sets .STAT to 1.

If you specify –BRIEF, most error messages produced by the function are suppressed. Only error messages indicating improper invocation of the function or its arguments are printed.

▶ [RESCAN string]

RESCAN removes one level of quotes from **string** and evaluates any function calls or variable references no longer appearing in quotes.

For example,

```
[RESCAN '[DATE -DOW]']
```

returns the day of the week.

The function may be used to force evaluation of quoted variables.

▶ [RESPONSE text [default] [–TTY]]

RESPONSE prints **text** on the terminal screen and follows it with a colon.

If **text** is null, printing is suppressed. **text** and **default** must be entered in single qoutes if they contain special characters or embedded blanks. One level of quoting is stripped before printing. After **text** has been printed, the function awaits your input. Your response is then returned as the value of the function, as in the following example:

```
OK, TYPE [RESPONSE HELLO]
HELLO: GOODBYE
GOODBYE
OK,
```

If the response is a carriage return, **default** is returned. If you omit **default**, the null string is returned.

The –TTY option forces the RESPONSE function to take input from the terminal. If you omit this option, the function takes its response from the command input stream. That is, if the function is invoked interactively, it goes to the terminal for its response. If the function is invoked from a command input file, or from a &DATA group within a CPL program, it goes to the command input file or CPL program for its input. A CPL program containing the RESPONSE function with the –TTY option cannot be executed as a batch job or phantom.

▶ [SEARCH string1 string2]

SEARCH returns the index of the first character in **string1** that appears in **string2**.

For example,

```
[SEARCH ABCDEFG XYEF]
```

returns

```
5
```

because E is the fifth character in the string abcdefg. If no character in **string1** appears in **string2**, SEARCH returns 0.

▶ [SUBST string1 string2 string3]

SUBST (substitute) substitutes **string3** for **string2** wherever **string2** occurs within **string1**, and returns the altered **string1**.

For example,

```
[SUBST ABCDEFG DE QZ]
```

returns

```
ABCQZFG
```

▶ [SUBSTR string start-position [num_chars]]

SUBSTR (substring) returns a substring of **string** that begins at position **start-position** and extends for length **num-chars**.

If you omit **num-chars**, SUBSTR prints all characters from **start-position** to the end of **string**. **start-position** and **num-chars** (if given) must be positive integers.

For example,

```
[SUBSTR ABCDEFGHIJ 4 3]
```

returns

```
DEF
```

▶ [TO_HEX number]

TO_HEX converts a decimal integer to its hexadecimal equivalent. Negative numbers are supported by this function.

number is the decimal number to be converted. For example,

```
[TO_HEX 12]
```

returns

```
C
```

▶ [TO_OCTAL number]

TO_OCTAL converts a decimal integer to its octal equivalent. Negative numbers are supported by this function.

number is the decimal number to be converted. For example,

```
[TO_OCTAL -8]
```

returns

```
-10
```

▶ [TRANSLATE string1 string2 string3]

TRANSLATE replaces characters in one string with characters from another.

TRANSLATE looks for **string3** characters in **string1**, replaces them with corresponding characters from **string2**, then returns the altered **string1**.

For example,

```
[TRANSLATE BRAVE 12345 ABCDEF]
```

returns

```
2R1V5
```

If **string2** and **string3** are omitted, TRANSLATE converts all **string1** characters to uppercase, then returns **string1**.

▶ [TRIM string [{ –LEFT / –RIGHT / –BOTH }] [char]]

TRIM removes a specified character from the left, the right, or both sides of a given string.

string is the string of characters to be modified, and **char** is the character to be removed. For example,

```
[TRIM XXHELLOXX -BOTH X]
```

returns

```
HELLO
```

If you omit **char**, leading and/or trailing blanks are removed. If you do not specify a side (–LEFT, –RIGHT, or –BOTH) when blanks are removed, –BOTH is assumed.

▶ [UNQUOTE string]

UNQUOTE removes the outer pair of quotes around the text specified in **string** and changes all other remaining pairs of quotes within **string** to single quotes.

For example,

```
[UNQUOTE '''XX''''YY''']
```

returns

```
'XX''YY'
```

Note that the TYPE command also removes a level of quotes from its argument, so that the following results are obtained when TYPE and UNQUOTE are used together:

```
OK, TYPE [UNQUOTE '''XX''''YY''']
XX'YY
OK,
```

▶ [VERIFY string1 string2]

VERIFY returns an integer that represents the position (beginning with 1) of the first character in **string1** that does not appear in **string2**.

For example,

```
[VERIFY 123X456Y 654321]
```

returns

```
4
```

because X, the fourth character of 123X456Y, is the first character that does not appear in 654321.

If all characters of **string1** appear in **string2**, VERIFY returns 0.

▶ [WILD wild-name-1 [...wild-name-n] [options] [–BRIEF]]

WILD produces a list of all names within a directory that match one or more wildcard names.

The WILD function has two forms, which are discussed below. The first form returns all matching names at once, in a single list. Names within the list are separated by blanks. The list may not exceed 1024 characters in length. The second form, which uses the –SINGLE option, returns one matching name per invocation until the list of names is exhausted.

wild-name-1 through **wild-name-n** are wildcard names that the WILD function matches. If **wild-name-1** is a pathname, all the wild-names are looked for in the directory that **wild-name-1** specifies. Otherwise, all names are searched for in the current directory. **wild-name-2** through **wild-name-n** may not be pathnames.

If, for example, the directory MYDIR contains the files LETTER1 through LETTER5 and the directories SUB1 and SUB2, then

```
[WILD MYDIR>LETTER@ SUB@]
```

returns

```
LETTER1 LETTER2 LETTER3 LETTER4 LETTER5 SUB1 SUB2
```

Note

The entries are not necessarily returned in alphabetical order.

options represent one or more optional control arguments that place limits on the objects matched by the specified wildcard names. **options** are as follows:

Option	*Meaning*
–ACCESSED_AFTER date	Matches objects in Rev. 20.0 or later directories last accessed on or after **date**. (One format of **date** is mo/dd/yy.hh:mm:ss. For complete information on how to specify **date**, see the section Wildcard Options in Chapter 4.)
–ACCESSED_BEFORE date	Matches objects in Rev. 20.0 or later directories last accessed before **date**.
–ACCESS_CATEGORY	Matches access categories only.
–AFTER date	Matches objects last modified on or after **date**.
–BACKEDUP_AFTER date	Matches objects backed up with the BACKUP utility on or after **date**.
–BACKEDUP_BEFORE date	Matches objects backed up with the BACKUP utility before **date**.
–BEFORE date	Matches objects last modified before **date**.
–CREATED_AFTER date	Matches objects in Rev. 20.0 or later directories created on or after **date**.
–CREATED_BEFORE date	Matches objects in Rev. 20.0 or later directories created before **date**.
–DIRECTORY	Matches directories only.
–FILE	Matches files only.
–MODIFIED_AFTER date	Same as –AFTER **date**.
–MODIFIED_BEFORE date	Same as –BEFORE **date**.
–RBF	Matches ROAM files only.
–SEGMENT_DIRECTORY	Matches segment directories only.

Note

Objects with undefined date/time values are matched by both members of each pair of date-selection options. Thus, an object with an undefined DTA value is matched by both –ACCESSED_AFTER and –ACCESSED_BEFORE. An object with an undefined DTC value is matched by both –CREATED_AFTER and –CREATED_BEFORE. Each pair in the –BEFORE and –AFTER families of options behaves in this way.

If you specify –BRIEF, most error messages produced by the function are suppressed. Only error messages indicating improper invocation of the function or its arguments are printed.

The second form of the WILD function returns names one at a time, rather than as a list. Its format is

[WILD wild-name-1 [...wild-name-n] [options] [–SINGLE unit-var] [–BRIEF]]

Use –SINGLE whenever you think the WILD list might overrun its limit of 1024 characters, or whenever it is more convenient to work with filenames one at a time.

Set **unit-var** to 0 before using the WILD function with the –SINGLE option. (**unit-var** must be a global variable if the function is invoked at command level.) WILD uses **unit-var** to store the number of the file unit on which it opens the directory for reading. The directory remains open until all matching names have been returned.

WILD returns the true null string if no entries are matched, or if, in –SINGLE mode, the end of the directory is reached.

Note

You can use all WILD options with any PRIMOS command that accepts wildcards: LD, COPY, DELETE, SIZE, and so on.

▶ [WRITE_FILE unit text]

WRITE_FILE writes the contents of **text** into a previously opened ASCII file.

WRITE_FILE returns 0 if the operation is successful and returns a positive integer otherwise. **text** must be entered in single quotes if it contains special characters. One level of quoting is stripped prior to writing.

unit is the decimal integer identifying the file to be written. It may be either the number returned by the OPEN_FILE function or a variable whose value was set to that number. (At command level, only global variables may be used. Both local and global variables are allowed within CPL programs.)

text is the information to be written. It is written as a single line in the file; that is, the newline is automatically added. If, for example, the file STUFF was opened on unit 37, then

```
[WRITE_FILE 37 'ADD THIS STUFF.']
```

writes the text to the file and returns a value of 0. If the variable .FILE_NUM was previously set to the file unit number, then

```
[WRITE_FILE %.FILE_NUM% 'ADD THIS STUFF.']
```

does the same thing.

If WRITE_FILE is used to write to a file that already contains **text**, the old text is overwritten by the new text. To add new text to a file without writing over the old text, use READ_FILE to move down to the end of the old text before using WRITE_FILE.

4 Command-line Features

Introduction

The PRIMOS command processor supports many command-line features that you can use to control and facilitate command processing. Among these features are the following:

- Global variables to supply information to PRIMOS commands and to user programs
- Multiple command entry to give several commands at once
- Iteration lists to repeat a command, substituting a new file system object each time
- Wildcard symbols to execute a command on a group of objects without listing their names individually
- Treewalking to search through a directory tree to execute a specified command on the appropriate file system objects
- Name generation to avoid repeating long entrynames by having PRIMOS substitute the full names for generation symbols
- Combinations of command-line features
- Syntax suppression to ignore special characters and command functions on the command line

You can also use the ABBREV command to create abbreviations for commands and command lines that you often use. For details on using ABBREV, see Chapter 2 of this book or the *PRIMOS User's Guide*.

Global Variables

Global variables are used in two different ways. In programs, global variables are used to define a variable that is known to more than one program. At the command-line level, global variables are often used to supply long pathnames to PRIMOS commands and variable arguments to certain command functions.

The DEFINE_GVAR command creates a global variable file; the SET_VAR command defines each global variable. Both of these commands, in addition to the DELETE_VAR and LIST_VAR commands, are explained in Chapter 2 of this guide and in the *PRIMOS User's Guide*.

Supplying Long Pathnames or Arguments

Global variables, like abbreviations created with the ABBREV command, can be used to supply long pathnames in a shortened form. When used as an argument to a command, a global variable must be specified with a percent sign (%) on each side.

For example, if you store source programs in the subdirectory BRANCH1 and their corresponding binary files in subdirectory BRANCH2, you could create a global variable for the complete pathname of each subdirectory:

```
OK, SET_VAR .SRC <FOREST>MAPLE>BRANCH1
OK, SET_VAR .BNY <FOREST>MAPLE>BRANCH2
```

To compile GRADES.FTN (a FORTRAN source file in BRANCH1) and store the resulting binary file in BRANCH2, you would give the command

```
OK, FTN %.SRC%>GRADES.FTN -B %.BNY%>GRADES.BIN
```

Supplying Variables to Command Functions

Certain command functions, such as OPEN_FILE and READ_FILE, require either a local or a global variable as an argument. You can supply such an argument with a predefined global variable. Do not use the percent sign (%) with command functions if the argument requires a variable. For further information, see the *CPL User's Guide* or Chapter 3 of this guide.

Multiple Commands

You can issue several commands on one command line if each command is separated by the command separator character, the semicolon (;). Each command is executed separately, as if it were on its own command line. For example, the following command line attaches you to the directory MARKET and then lists its contents:

```
ATTACH MARKET; LD
```

If one of the commands contains an error, PRIMOS still attempts to execute the remaining commands on the line. For instance, if PRIMOS is unable to attach to MARKET (in the previous example), it still lists the contents of whatever directory you are attached to.

The ABBREV command disables the effect of semicolons as command separators and interprets them literally, as part of the abbreviation rather than as a separator between ABBREV and another command. Therefore, if you give an ABBREV command followed by another command on the same line, the second command will become part of the preceding abbreviation. This feature allows you to create an abbreviation that executes more than one command. For example, the command

```
ABBREV -ADD_COMMAND CA CLOSE -ALL; RLS -ALL
```

defines an abbreviation that, when executed, closes all your files and then releases the entire stack.

The effect of semicolons is also disabled if the command line begins with a tilde (~), the syntax suppression character.

Note

Do not use the COMINPUT command on a command line that contains multiple commands.

Iteration

There are occasions when you must repeat the same command several times, specifying a new file (or other file system object) each time. For example, to compile three FORTRAN files, you must give the following three commands:

```
FTN A -64V -XREF
FTN B -64V -XREF
FTN C -64V -XREF
```

You can do this more easily by giving one command that contains the three filenames enclosed in parentheses. Thus, the previous three commands can be given as one:

```
FTN (A B C) -64V -XREF
```

The list in parentheses is called an **iteration list**. Each member of the iteration list must be separated by a blank or by a comma. The previous example can also be given as

```
FTN (A,B,C) -64V -XREF
```

PRIMOS executes the command once for each member of the list, as if you had typed each command separately.

Note

To separate items within an iteration list, use spaces, commas, or a combination of a comma followed by a space. You cannot use the combination of a space followed by a comma.

Multiple Iteration Lists

The previous example replaced one argument with one iteration list. You can, however, replace as many arguments as you like with iteration lists. You must separate each list with a blank or a comma. For example, the command

```
COPY  (A B C)  (D E F)
```

copies files A, B, and C, renaming them D, E, and F, respectively. The iteration list (A B C) is the first argument and the list (D E F) is the second.

When an iteration list runs out of arguments, a null string is substituted. For example, the command

```
CMPF (L M N)  (O P)  (Q R S)
```

produces the following three commands:

```
CMPF  L O Q
CMPF  M P R
CMPF  N S
```

Iteration Lists as Parts of Arguments

The previous examples used iteration lists as complete arguments. An iteration list, however, can be used as part of a single argument if the list is not separated by a blank or a comma from the rest of the argument. For example, the following command uses an iteration list as part of each of the two arguments:

```
COPY ALLEN>(A B C) JUNE>(D E F)
```

When executed, the command copies the files A, B, and C from the directory ALLEN into the directory JUNE, naming them D, E, and F, respectively.

No more than two iteration lists can be used in a single argument.

Cross-product Iteration Lists

When you use two (but no more than two) iteration lists within a single argument, you create a **cross-product iteration list**. In such a list, each member of the first iteration list is paired with each member of the second iteration list, and the command is executed once for each object that results. For example, the command

```
DELETE (YOUR,MY,HIS).(MEMO,DRAFT)
```

uses one argument consisting of two iteration lists and produces a cross-product iteration list that deletes the following files: YOUR.MEMO, YOUR.DRAFT, MY.MEMO, MY.DRAFT, HIS.MEMO, and HIS.DRAFT.

Similarly, the command

```
DELETE (A B C) (D E F)
```

deletes the following nine files: AD, AE, AF, BD, BE, BF, CD, CE, and CF.

Wildcarding

Wildcarding allows you to specify groups of file system objects on which a command can act. For example, the following command compiles all files in the current directory whose names have .FTN as a suffix:

```
FTN @@.FTN
```

The following command lists all file system objects in the directory BEECH whose names begin with the letter A.:

```
LD BEECH>A@@
```

Wildcarding is specified by using a wildcard name as an argument to a command. A **wildcard name** is a pathname in which the final element (or the only element, if the name is an entryname) contains one or more of the wild characters shown in Table 4-1.

TABLE 4-1
Wild Characters

Character	*Function*
@@	Replaces any number of characters in any number of components within a file or directory name.
@	Replaces any number of characters within one component of a filename or directory name. Stops matching at the period (.) that separates a name and its suffix.
+	Replaces a single character, except a period(.).
^	Inverted match character. Must be the first character in the wildcard name. A wildcard name that begins with ^ matches all names that do not match the rest of the wildcard name.

Examples of each of these characters are given in subsequent sections.

In most cases, only one argument per command can contain wildcard names. That argument can contain either a single wildcard name (as in the examples above) or a single iteration list containing any number of wildcard names, as in the following examples:

```
DELETE (A@@ B@@)

DELETE @@(.BIN .LIST)
```

Wildcard Matching

When a command containing a wildcard name is given, the command processor searches the specified directory for all entrynames that match the given wildcard name. An entryname matches a wildcard name when it contains both of the following:

- The same number of components as the wildcard name
- The same literal characters as the wildcard name contains, in the same relative positions

The three wildcard symbols +, @, and @@ differ only in the number of characters they can match.

- + matches only one character (except a period).
- @ matches any number of characters in a single component of a name, stopping at the period (.) that separates components.
- @@ matches any number of characters in any number of components.

Thus, a wildcard name consisting only of the symbol @ matches all single-component names in the directory, whereas a wildcard name consisting only of the symbol @@ matches all names in the directory, regardless of how many components they contain.

The selection order for wildcard matching cannot be predicted.

Examples of Wildcard Names

Suppose that the current directory contains the following file system objects:

```
BARR1.COBOL     BARR1.SEG     BARR2.COBOL     BARR2.SEG
CLR.CPL         EDD.COMO      EDD.COMO.OLD    EDD.CPL
EDD.CPL.OLD     FILL          SCROLL          SKILL
```

The wildcard name @ matches all single-component names:

```
FILL            SCROLL        SKILL
```

The wildcard name S@ matches all single-component names that begin with S:

```
SCROLL          SKILL
```

The wildcard name @.@ matches all two-component names:

```
BARR1.COBOL     BARR1.SEG     BARR2.COBOL     BARR2.SEG
CLR.CPL         EDD.COMO      EDD.CPL
```

The wildcard name @.SEG matches all two-component names that end in .SEG:

```
BARR1.SEG       BARR2.SEG
```

The wildcard name @.@.@ matches all three-component names:

```
EDD.COMO.OLD    EDD.CPL.OLD
```

The wildcard name BARR+.COBOL matches

```
BARR1.COBOL     BARR2.COBOL
```

The wildcard name BARR+.@ matches

```
BARR1.COBOL     BARR1.SEG     BARR2.COBOL     BARR2.SEG
```

The wildcard name @@L matches all names, of any number of components, that end in L:

```
BARR1.COBOL     BARR2.COBOL   CLR.CPL         FILL
EDD.CPL         SCROLL        SKILL
```

The wildcard name @@ matches every name in the directory.

Inverted Matching

The caret (^) functions as an inverted match character. Any wildcard name beginning with a caret matches every object whose name does not match the rest of the wildcard name. Using the directory above, the wildcard name ^@@L matches all names that do not end in L:

```
BARR1.SEG       BARR2.SEG     EDD.COMO        EDD.COMO.OLD
EDD.CPL.OLD
```

Wildcard Options

You can modify the effect of wildcard names by specifying one or more of the wildcard options described in Table 4-2. **Wildcard options** can be given on the command line anywhere after the command.

Wildcard options are used to

- Select only file system objects of a particular type or types
- Select only file system objects that were last modified before or after a particular date
- Enable or disable verification

TABLE 4-2
Wildcard Options

Option	*Selects*
–ACCESSED_AFTER date	Objects in Rev. 20.0 or later directories last accessed on or after **date.**
–ACCESSED_BEFORE date	Objects in Rev. 20.0 or later directories last accessed before **date.**
–ACCESS_CATEGORY	Access categories.
–AFTER date	Objects last modified on or after **date.**
–BACKEDUP_AFTER date	Objects saved by BACKUP on or after **date.**
–BACKEDUP_BEFORE date	Objects saved by BACKUP before **date.**
–BEFORE date	Objects last modified before **date.**
–CREATED_AFTER date	Objects in Rev. 20.0 or later directories created on or after **date.**
–CREATED_BEFORE date	Objects in Rev. 20.0 or later directories created before **date.**
–DIRECTORY	MFDs, directories, and subdirectories.
–FILE	SAM or DAM files.
–MODIFIED_AFTER date	Same as –AFTER.
–MODIFIED_BEFORE date	Same as –BEFORE.
–NO_VERIFY	Suppresses verification requests, even for a command that usually requires it. (Default)
–RBF	ROAM files.
–SEGMENT_DIRECTORY	SAM or DAM segment directories.
–VERIFY	Causes PRIMOS to request verification before executing the command on any object.

The options are discussed in the following three sections. The formats for specifying **date** are discussed in the second section.

Type-designation Options: The five options that select particular types of file system objects are the following:

–ACCESS_CATEGORY
–DIRECTORY
–FILE
–RBF
–SEGMENT_DIRECTORY

When you specify one or more of these options in a command line containing a wildcard name, the command executes only on objects of that type. For example, the following command lists all files beginning with A in the current directory:

```
LD A@@ -FILE
```

The following command lists all files and access categories beginning with A in the current directory:

```
LD A@@ -FILE -ACCESS_CATEGORY
```

If you specify none of the type-specification options, all file object types (files, directories, segment directories, and access categories) are matched.

Date-selection Options: The following options select an object according to the date on which it was created, last modified, last accessed, or backed up:

–ACCESSED_AFTER date
–ACCESSED_BEFORE date
–AFTER date
–BEFORE date
–CREATED_AFTER date
–CREATED_BEFORE date
–MODIFIED_BEFORE date
–MODIFIED_AFTER date
–BACKEDUP_BEFORE date
–BACKEDUP_AFTER date

When you specify these options, the command processor matches only objects that were created, last modified, last accessed, or backed up before, on, or after a given date and time.

The four full formats with which you specify **date** are

mo/dd/yy.hh:mm:ss
yy-mo-dd.hh:mm:ss
'dd mon yy.hh:mm:ss'
'dd mon yy hh:mm:ss'

where **mon** stands for the first three characters of a month (JAN, FEB, MAR, etc.) and all other symbols are one-digit or two-digit numbers that represent the following:

Symbol	*Meaning*
yy	Year (19yy)
mo	Month (January = 1 or 01, and so on)
dd	Day (1 or 01 to 31)
hh	Hour (24-hour designation)
mm	Minute
ss	Second

Not all items in the full date formats are required. The following defaults are assumed when items are omitted:

Omitted Item	*Default Value*
mo/dd/yy **yy-mo-dd** **dd mon yy**	Today's date
yy	Today's year
hh:mm:ss	00:00:00 (the beginning of the specified day)
mm	00
ss	00

The following are examples of valid dates and times:

Example	*Meaning*
`-AFTER 03/17/87.13:23:09`	On or after nine seconds past 1:23 PM, March 17, 1987
`-BEFORE 87-3-17.13:23:9`	Before nine seconds past 1:23 PM, March 17, 1987
`-AFTER '17 MAR 87 13:23:9'`	On or after nine seconds past 1:23 PM, March 7, 1987
`-AFTER 11:15:35`	On or after 11:15 AM, plus 35 seconds, today's date
`-BEFORE 1/14/87`	Before January 14, 1987
`-AFTER 2-22`	On or after midnight as 21 Feb turns to 22 Feb
`-BEFORE 16`	Before 4 PM, today's date
`-AFTER 3.`	On or after the beginning of the 3rd day, current month and year (note the period after the 3)
`-BEFORE '7 MAR.11'`	Before 11 AM, on 7 March, current year
`-AFTER 0`	Everything on or after midnight of last night

The presence or absence of the period separator can determine how the date and time are interpreted, as shown in the following example:

Designation	*Meaning*
`-BEFORE 10`	Before 10 AM of today's date
`-BEFORE 10.`	Before the 10th day of the current month and year

If both –BEFORE and –AFTER options are given, an object must satisfy both options in order to be selected. For example, the following command lists only segment directories that were last modified between December 15, 1986 and March 1, 1987:

```
LD @@ -SEGDIR -AFTER 12/15/86 -BEFORE 3/1/87
```

This rule also applies to the –MODIFIED_BEFORE/–MODIFIED_AFTER and –BACKEDUP_BEFORE/–BACKEDUP_AFTER options.

Verification Options: The two verification options are –VERIFY and –NO_VERIFY. These options differ from other wildcard options in that they do not control the actual matching of objects. Rather, they enable or disable verification for the commands that act on the chosen objects.

Some commands, such as DELETE, produce results that are irrevocable. When you issue such a command a wildcard name, PRIMOS asks you for verification before each execution of the command. This ensures that you really want the action performed on that particular object.

For example, if you have the files BAT, CAT, and RAT in the current directory, you would give the following command and responses to delete BAT and RAT:

```
OK, DELETE @@AT
(std$cp) Verify wildcard selections for "@@AT":
"BAT"? YES
"CAT"? NO
"RAT"? YES
OK,
```

When PRIMOS asks for a verification for a name, you respond with one of the following four answers:

- YES or Y to execute the command on the name after all names have been verified.
- NO, N, or a carriage return to skip execution on the name.
- NEXT to cancel execution for all names, even those previously verified with YES. This cancelation does not affect any iteration or treewalking in progress.
- Pressing the BREAK or CONTROL-P keys to end all processing of the command line and return to PRIMOS command level.

By using the –NO_VERIFY option, you can disable verification for a command that normally requests it. For example, if you do not want to be asked to verify deletions, give the above DELETE command as follows:

```
DELETE @@AT -NO_VERIFY
```

The –VERIFY option allows you to enable verification for a command that normally does not require it.

Note

The wildcard –NO_VERIFY option is different from the –NO_QUERY option of some PRIMOS commands, such as DELETE. Specifying –NO_QUERY with a wildcard name still results in verification requests because the wildcard verification default overrides the –NO_QUERY option.

Treewalking

Wildcarding allows PRIMOS commands to act on a group of file system objects located within a single directory. **Treewalking** takes this convenience one step further, and allows a command to act on designated objects within a directory tree (that is, a directory, its subdirectories, their subdirectories, and so forth).

You specify a treewalking pattern by using wildcard characters in an intermediate position within a pathname and/or in the final position of the pathname. The wildcard characters cannot be in the first position of the pathname.

When you give a command that contains a treewalk name, the command processor searches all directories subordinate to the specified starting directory for file system objects that match the given treewalk pathname. (The starting directory itself is not searched unless you specify the –WALK_FROM 1 option, explained below.)

Figure 4-1 illustrates how PRIMOS proceeds vertically through directories in a standard treewalk. The horizontal order of visitation within a single directory cannot be predicted. Hence, the horizontal order of the top three subdirectories (numbers 1, 6, and 8) in Figure 4-1 might differ in a subsequent treewalk. The command for Figure 4-1 is

```
LD ORCHARD>@@>@@
```

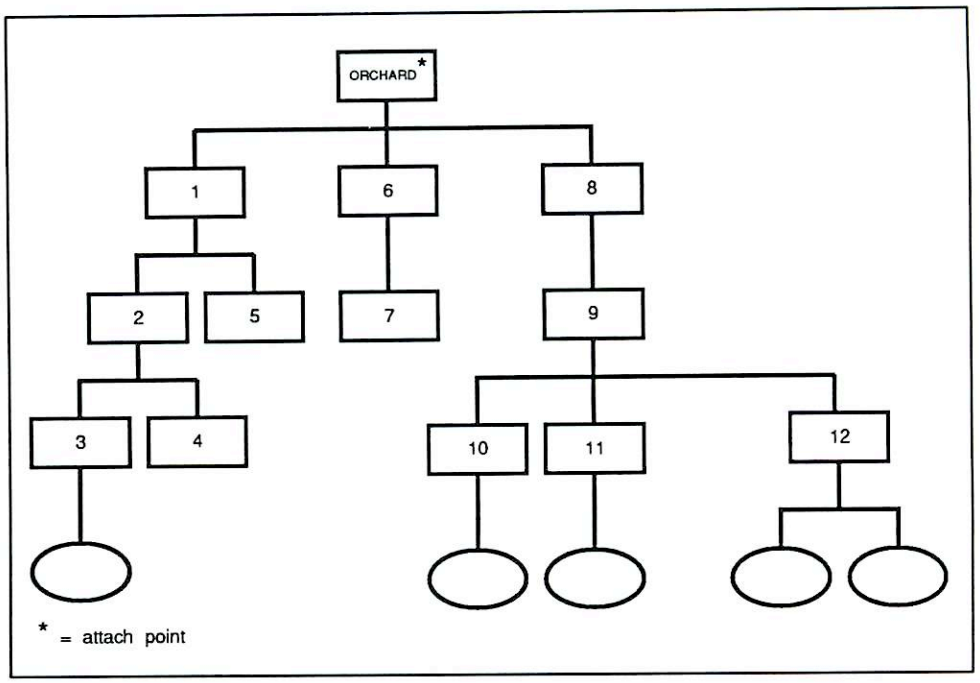


FIGURE 4-1
Order of Visiting Directories in Sample Standard Treewalk

Examples of Treewalking

Figure 4-2, representing a sample directory tree stemming from the directory ORCHARD, is used for the following examples.

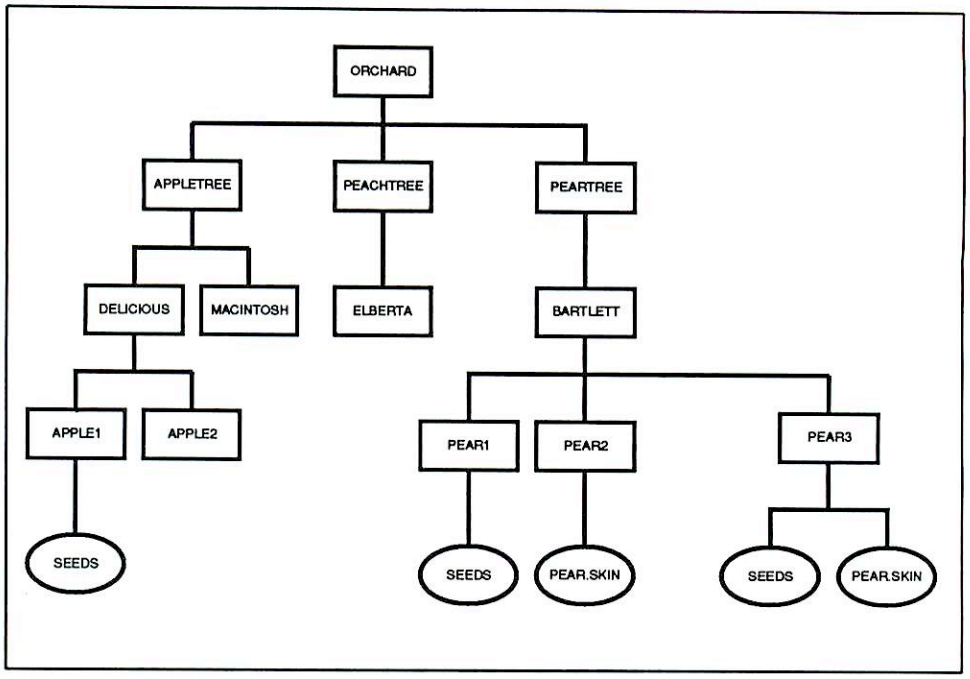


FIGURE 4-2
Sample Directory Tree

If you are attached to directory ORCHARD, the command

```
LD @@ -DIR
```

produces the following list of all the directories contained in ORCHARD:

```
OK, LD @@ -DIR

<FOREST>ORCHARD (ALL access)
1 record in this directory, 18 total records out of quota of 0.

3 Directories.

APPLETREE          PEACHTREE          PEARTREE
```

(The command could also be given as LD –DIR because @@ is assumed when no pathname follows the LD command.)

To list all subdirectories subordinate to ORCHARD, issue the command

```
LD ORCHARD>@@>@@ -DIR
```

This displays information on all subdirectories from APPLETREE to PEAR3, as shown in Figure 4-3.

```
OK, LD ORCHARD>@@>@@ -DIR

<FOREST>ORCHARD>APPLETREE (ALL access)
1 record in this directory, 6 total records out of quota of 0.

2 Directories.

DELICIOUS MACINTOSH


<FOREST>ORCHARD>APPLETREE>DELICIOUS (ALL access)
1 record in this directory, 4 total records out of quota of 0.

2 Directories.

APPLE1 APPLE2


<FOREST>ORCHARD>APPLETREE>DELICIOUS>APPLE1 (ALL access)
2 records in this directory, 2 total records out of quota of 0.

No entries selected.


<FOREST>ORCHARD>APPLETREE>DELICIOUS>APPLE2 (ALL access)
1 record in this directory, 1 total record out of quota of 0.

No entries selected.


<FOREST>ORCHARD>APPLETREE>MACINTOSH (ALL access)
1 record in this directory, 1 total record out of quota of 0.

No entries selected.
```

FIGURE 4-3
Sample Treewalking Terminal Display

```
<FOREST>ORCHARD>PEACHTREE (ALL access)
1 record in this directory, 2 total records out of quota of 0.

1 Directory.

ELBERTA


<FOREST>ORCHARD>PEACHTREE>ELBERTA (ALL access)
1 record in this directory, 1 total record out of quota of 0.

No entries selected.

<FOREST>ORCHARD>PEARTREE (ALL access)
1 record in this directory, 9 total records out of quota of 0.

1 Directory.

BARTLETT


<FOREST>ORCHARD>PEARTREE>BARTLETT (ALL access)
1 record in this directory, 8 total records out of quota of 0.

3 Directories.

PEAR1 PEAR2 PEAR3


<FOREST>ORCHARD>PEARTREE>BARTLETT>PEAR1 (ALL access)
2 records in this directory, 2 total records out of quota of 0.

No entries selected.


<FOREST>ORCHARD>PEARTREE>BARTLETT>PEAR2 (ALL access)
2 records in this directory, 2 total records out of quota of 0.

No entries selected.


<FOREST>ORCHARD>PEARTREE>BARTLETT>PEAR3 (ALL access)
3 records in this directory, 3 total records out of quota of 0.

No entries selected.

OK,
```

FIGURE 4-3
Sample Treewalking Terminal Display - Continued

You could also list all the directories in the subtree headed by directory APPLETREE by issuing the command

```
LD ORCHARD>APPLETREE>@@>@@ -DIR
```

To locate all files named SEEDS in the directory tree, give the command

```
LD ORCHARD>@@>SEEDS
```

To list all files ending in .SKIN in the subtree headed by directory PEARTREE, issue the command

```
LD ORCHARD>PEARTREE>@@>@@.SKIN
```

These examples use the @@ wildcard character, but you can use any wildcard characters in a treewalk name. When used in treewalk names, the wildcard characters have the same meanings that they have in wildcard names.

Treewalking Options

You can change the order of the search or the number of directories visited by using the treewalking options explained in Table 4-3.

TABLE 4-3
Treewalking Options

Option	*Meaning*
–WALK_FROM n	Executes the command in directories at levels greater than or equal to n. The default is –WALK_FROM 2, which executes the command at the first directory under the starting directory. For execution in the starting directory, specify –WALK_FROM 1.
–WALK_TO n	Executes the command in directories at levels less than or equal to n.
–BOTTOM_UP	Executes the command in specified directories starting at the deepest level (that is, starting at the largest level number and going to the smallest level number). The default is to start at the highest level (that is, the smallest level number) and work down.

Examples of Treewalking Using Options

Figure 4-4 illustrates the order of visiting directories in a bottom-up treewalk in the sample tree ORCHARD. The command is

```
LD ORCHARD>@@>@@ -BOTUP
```

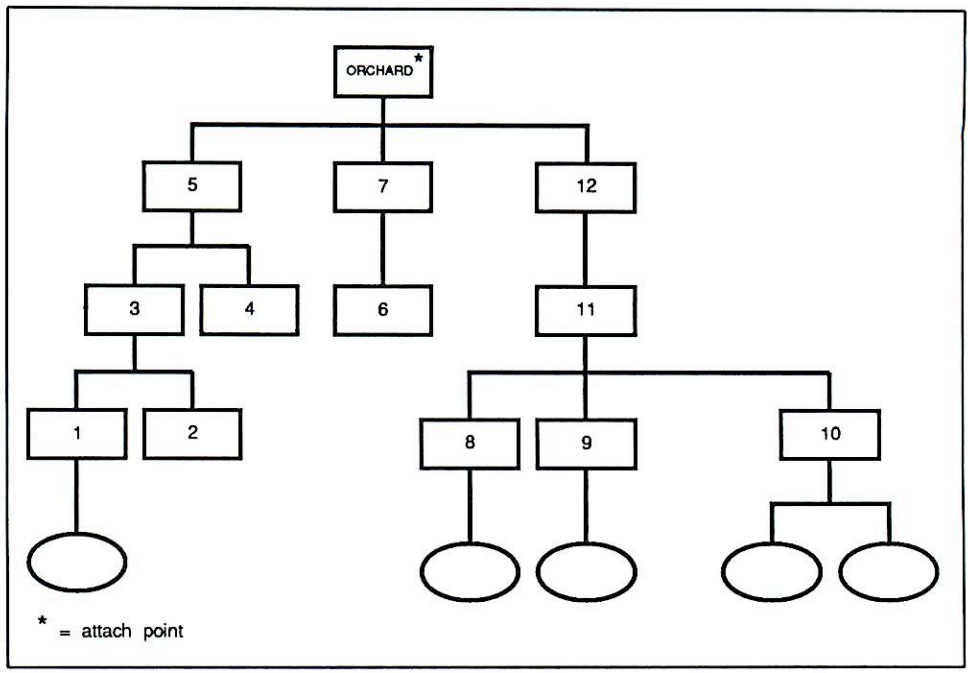


FIGURE 4-4
Order of Visiting Directories in Sample Bottom-up Treewalk

Figure 4-5 illustrates the order of visiting directories for a treewalk that includes the current directory (ORCHARD) but not the lowest directory level. The command is

```
LD ORCHARD>@@>@@ -WALK_FROM 1 -WALK_TO 3
```

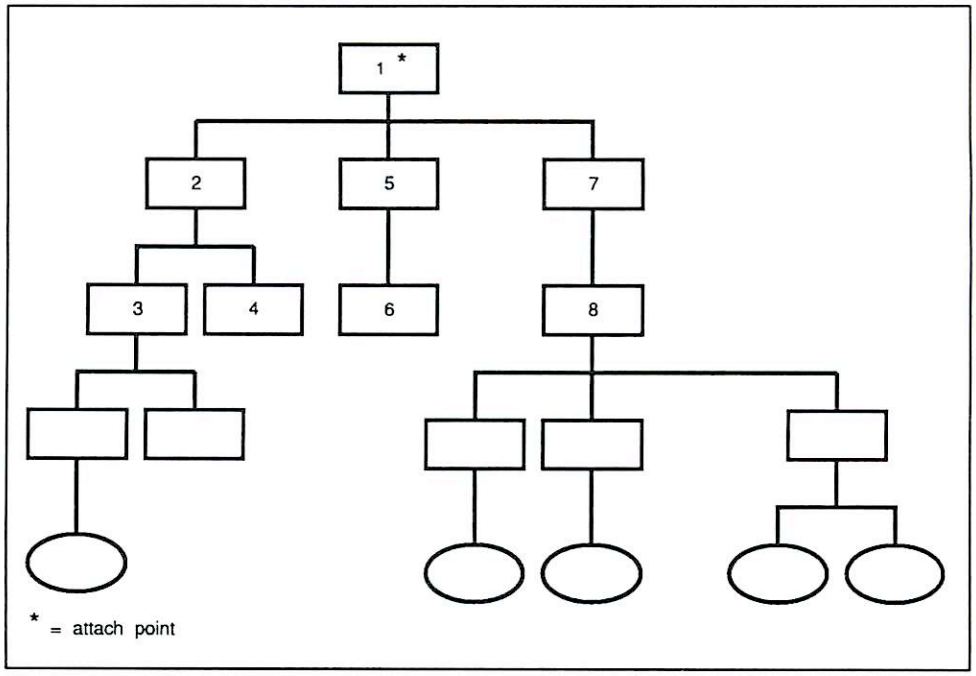


FIGURE 4-5
Effect of Specifying Levels to Treewalking Command

Name Generation

Name generation is a useful feature when you have to

- Specify several entrynames that are nearly identical.
- Create a group of names that parallel a group of wildcard names, as when copying and renaming a group of files.

In these cases, you can use name generation to quickly generate one or more pathnames. For example, you can do a file merge with the command

```
MRGF FILE.1 =.2 =.3 -OUTF =.4
```

This command compares FILE.1 against FILE.2 and FILE.3, and creates FILE.4 as a merged output file.

Requirements for Name Generation

Generated names are created from the following sources:

- One **source pathname**, from which to create new names. The number of components in the generated name is usually less than or equal to the number of components in the source pathname.

- **Generation patterns**, contained in the entryname portions (that is, the final positions) of one or more subsequent pathnames.

Source Pathname

Except for RESUME and SEG commands, the source pathname is the first pathname in the command line. That is, it forms the first argument to the command. For example, in the command line

```
CMPF DIR>NAMES.OLD DIR>NAMES.NEW
```

the source pathname is DIR>NAMES.OLD.

For RESUME and SEG commands, the source pathname is the second argument in the command line. For example, in the command line

```
RESUME MYPROGRAM DIR>ARGUMENT1 DIR>ARGUMENT2
```

the source pathname is DIR>ARGUMENT1.

Generation Patterns

Name generation patterns are composed of two items:

- Name generation symbols, usually the equal (=) and double-equal (==) signs. The equal sign (=) copies a single component of the source name; the double-equal sign (==) copies as many components as can be copied without adding components to the name. Only one double-equal sign can appear in a name generation pattern.
- Literal strings of characters, each of which replaces a component in the source name.

A summary of name generation symbols and their effects is shown in Table 4-4.

TABLE 4-4
Name Generation Symbols and Their Effects

Command	*Effect*
=	Copies a single component from the source name to the generated name. (Parallels the wild character @ for copying.)
= =	Copies one or more components from the source name to the generated name. (Parallels the wild character @@ for copying, except for such substitutions, additions, or deletions as are specified.)
^=	Skips over a single component from the source name without copying it to the generated name.
^= =	Skips over one or more components of the source name without copying them to the generated name.
literal-string	Replaces a component from the source name with the component given by **literal-string**.
+**literal-string**	Adds to the source name the component shown after the plus sign (+).

Examples of Name Generation

Source Name	*Generation Pattern*	*Generated Name*	*Comments*
`A.B.C.D`	`==.X`	`A.B.C.X`	Three components are copied to create a four-component name.
`A.B.C.D`	`==.X.Y`	`A.B.X.Y`	Only the first two components are copied.
`A.B.C.D`	`X.==.X`	`X.B.C.X`	Only the middle two components are copied.
`A.B.C.D`	`X.=`	`X.B`	A single equal sign copies only one component.
`A.B.C.D`	`=.X.Y.Z.=`		An error message results because the pattern specifies five components and the source name contains only four.

Note

A name generation pattern can be used only in the entryname portion (that is, the final position) of the pathname. The following command is therefore illegal:

```
COPY A>LONGNAME>B X>=>Y
```

Adding Components

To add a component to a generated name, precede the new characters or name generation symbols with a plus sign (+), as in the following examples:

Source Name	*Generation Pattern*	*Generated Name*
`A.B`	`==.+C`	`A.B.C`
`A.B`	`=.+C.=`	`A.C.B`
`A.B`	`+C.==`	`C.A.B`

The exception to this rule occurs when you are adding a component to the end of a name and have specified each preceding component explicitly. In this case, you specify the new component literally without using the plus sign, as in the following example:

Source Name	*Generation Pattern*	*Generated Name*
`A.B`	`=.=.C`	`A.B.C`

Deleting Components

To delete one or more components, precede the equal or double-equal sign with a caret (^), which acts as a negation symbol. The examples below show the creation of generated names with deleted components:

Source Name	*Generation Pattern*	*Generated Name*
A.B.C	=.^==	A
A.B.C	^=.==	B.C
A.B.C	=.^=.=	A.C
A.B.C	^==.=	C

Combining Additions, Substitutions, and Deletions

You can specify additions, substitutions, and deletions in a single name generation pattern, as in the following example:

Source Name	*Generation Pattern*	*Generated Name*
A.B.C.D.E.F	=.X.+Y.=.^=.=	A.X.Y.C.E

However, only one double-equal sign, with or without a caret (negation) sign, can appear in the pattern.

Combining Command-line Features

Some command-line features can be used in combination with others. The use of wild characters in treewalking is such an example. The next two sections discuss other combinations.

Combining Iteration With Other Features

Wildcards, treewalk patterns, and name generation patterns, as well as abbreviations, variables, and function calls, can all be used within iteration lists. For example, the following are all legal commands:

```
DELETE (A@@ B@@ C@@)
COPY (@.LIST @.BIN) ARCHIV>(=.OLDLIST =.OLDBIN)
SPOOL (%.YESTERDAY% %.TODAY% A.@@)
```

Wild Characters and Name Generation

If a source pathname includes a wildcard name, the generated names match whatever names are produced by the wildcarding process. For example, the command

```
COPY ALPHA>@@.LIST ARCHIV>==.OLDLIST
```

copies all listing files in the directory ALPHA into the directory ARCHIV and changes the suffix on each file from LIST to OLDLIST. The following three commands are among those that might be generated:

```
COPY  ALPHA>A.LIST ARCHIV>A.OLDLIST
COPY  ALPHA>B.LIST ARCHIV>B.OLDLIST
COPY  ALPHA>C.LIST ARCHIV>C.OLDLIST
```

Syntax Suppression

The tilde (~) is the PRIMOS syntax suppression character. When you begin a command line with a tilde, you force PRIMOS to interpret the rest of the line literally and ignore special characters, such as the wild, iteration, and name generation characters. Because these special characters initiate the command-line features described in this chapter, the features are therefore suppressed. Command functions (which use brackets as special characters) are also suppressed.

For example, the following command line executes both the TYPE and LD commands, thus printing the answer 200 and then listing the contents of your current directory:

```
TYPE [CALC 10 * 20];LD
```

If, however, you use a tilde as the first character of the command line, PRIMOS suppresses the special characters (that is, brackets and semicolon) and prints everything literally after TYPE, as follows:

```
OK, ~ TYPE [CALC 10 * 20];LD
[CALC 10 * 20];LD
OK,
```

Syntax suppression is particularly useful when you are creating an abbreviation that includes a command function or command-line feature. For example, in the following command, the DATE function is executed and the abbreviation LOG would stand for COMO LOG.MAY (if the current month were May):

```
ABBREV -AC LOG COMO LOG.[DATE -MONTH]
```

If, however, you want the DATE function inserted into the abbreviation without being evaluated, then you give the command

```
~ ABBREV -AC LOG COMO LOG.[DATE -MONTH]
```

The abbreviation LOG now stands for COMO LOG.[DATE –MONTH]. Thus, each time you issue the command LOG, the DATE function is evaluated and a command output file is created with a name beginning with LOG and having the current month as its suffix.

5 Command-line Processing

Introduction

If you frequently use the many features of the PRIMOS command processor, you may find it useful to know the order in which the features are processed and the interactions between them.

Command-line Processing Order

Command-line processing proceeds in the following order:

1. Expanding abbreviations
2. Suppressing syntax
3. Processing multiple commands
4. Evaluating variables and functions
5. Identifying iteration lists
6. Implementing treewalking
7. Matching wildcard names
8. Matching name generation patterns
9. Executing the command

Figure 5-1 illustrates this processing order. Each step is discussed in the following sections.

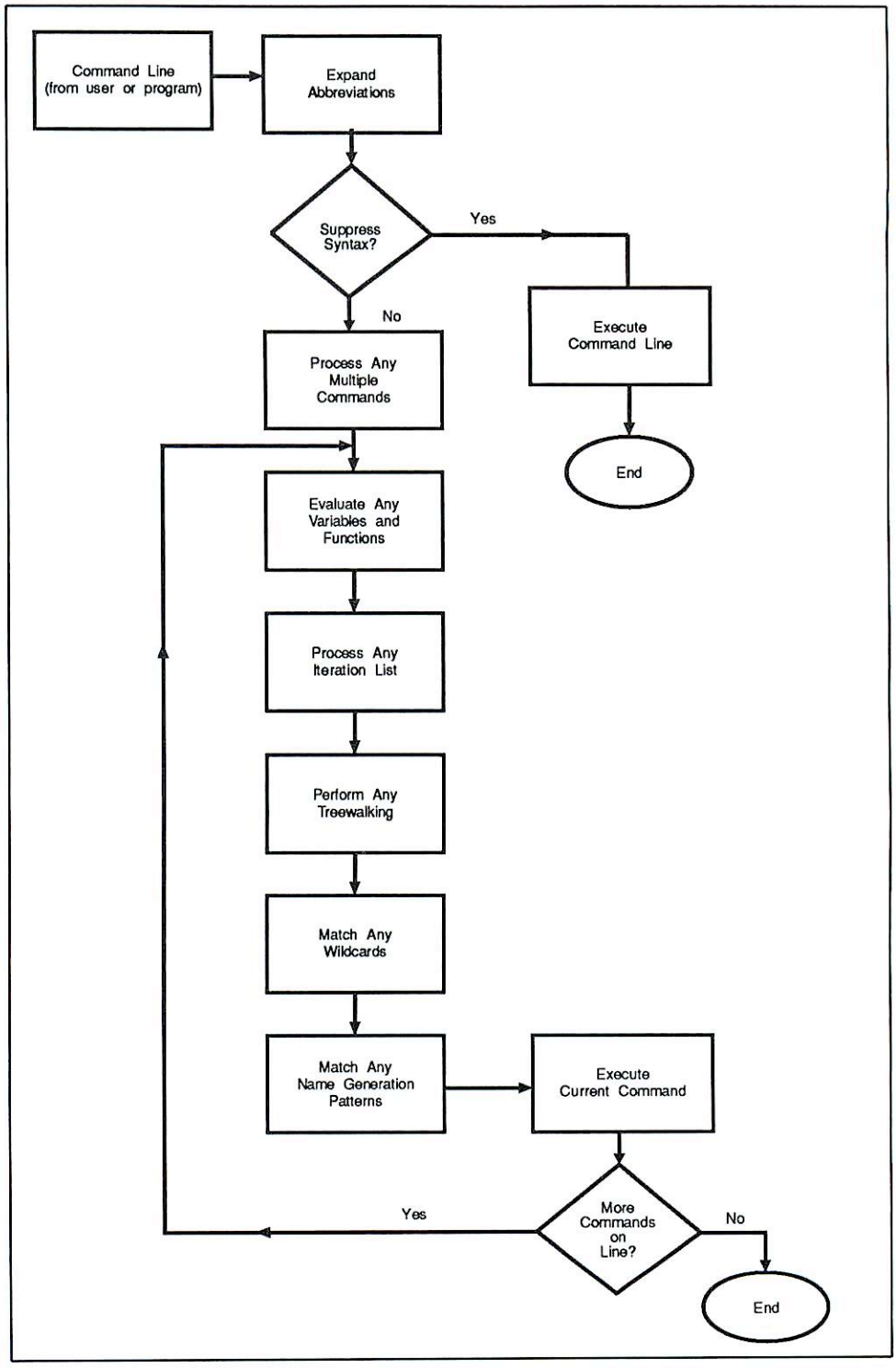


FIGURE 5-1
Process Flow for Execution of a Command Line

Expanding Abbreviations

If the System Administrator has not disabled the abbreviation processor and if your abbreviations file is active, the first thing that happens in the command line is that abbreviations are expanded.

A command function or simple iteration list on the command line is treated as a single token by the abbreviation processor. Functions or iteration lists that require abbreviation parameters or that are used as arguments for an abbreviation parameter are treated as single tokens by the abbreviation processor. For example, if A is the abbreviation for *>SUBDIR>%1%, then

```
A (B C D)
```

expands to

```
*>SUBDIR>(B C D)
```

because the list (B C D) is treated as a single token and is assigned as the value of parameter 1 of abbreviation A.

Similarly, if B is the abbreviation for *>BDIR>%1% and C is the abbreviation for *>CDIR>%1% then

```
(A B C) D E F
```

expands to

```
(*>SUBDIR>D *>BDIR>D *>CDIR>D) E F
```

because the list (A B C) is treated as a single token having one parameter, which is assigned the value D.

Function calls are also treated as single tokens, but they are expanded as if they were a separate command line. Thus, the first token following a left bracket is considered to be in the command position, and no abbreviation parameters are taken from beyond the matching right bracket.

For example, if A is the abbreviation for FOO %2%.TWO %1%.ONE, then

```
[A B] C D
```

expands to

```
[FOO .TWO B.ONE] C D
```

Note that C does not become the second parameter of A because C lies outside the function call brackets.

Suppressing Syntax

Following abbreviation expansion, the command processor checks to see if the first character on the command line is the syntax suppressor, the tilde (~). If so, processing of all subsequent command line features is suppressed. The tilde is removed and the remaining command line is executed as is. See Syntax Suppression in Chapter 4 of this guide for examples of using syntax suppression.

Processing Multiple Commands

The command processor next scans the command line for the command separator character, the semicolon (;). This character delimits multiple commands on the same command line. The recognition of the command separator is disabled if the syntax suppressor was used or if the command is ABBREV or AB. The latter exception is provided to allow definitions of abbreviations whose value contains the command separator character. For example,

```
ABBREV -AC ZOT CLOSE ALL; DELETE @@ -NO_VERIFY
```

is a single command that defines an abbreviation, ZOT, whose value is

```
CLOSE ALL; DELETE @@ -NO_VERIFY
```

If recognition of the command separator is not disabled, the features described hereafter are executed separately for each command on the command line. For example, the order of execution in the command line

command-1 [function-1 arguments]; command-2 [function-2 arguments]

is evaluate **function-1**, execute **command-1**, evaluate **function-2**, and execute **command-2**.

Note

Because of its operation, the COMINPUT command must not be used in command lines containing multiple commands.

Evaluating Variables and Functions

Once the current command has been identified, variable references are evaluated. Each reference of the form **%variable_name%** is replaced by the value of **variable_name**.

Text, command function references of the form **[function arguments]** are evaluated and replaced by their values. Evaluation proceeds from the inside out. For example, the command line

command [function arguments]

is replaced by

command value

where **value** is the value returned by **function**. (For details on functions, see the *CPL User's Guide* or Chapter 3 of this guide.)

The fact that variables and functions are evaluated after the command separator has been processed means that semicolons are not recognized as command separators when they form part of the value of a variable or function.

The fact that functions are evaluated after variables means that, if the value of a variable contains a function reference, the function will be evaluated. Conversely, if the value of a function contains a variable reference, the variable reference will not be evaluated because variable evaluation has already occurred.

If any error occurs during variable or function evaluation, such as a reference to an undefined variable or function, the command processor prints an error message and does not process that command.

Identifying Iteration Lists

All iteration sets (lists in parentheses) in the command line are identified. Conceptually, the command processor can be thought of as producing a series of command lines, one for each iteration specified by the simple iteration sets. For example,

```
command (A B C) Y (D E F)
```

can be thought of as the series of commands

```
command A Y D
command B Y E
command C Y F
```

In fact, the command processor does not generate the command strings at this time. Rather, it implements a kind of list structure at this level.

Implementing Treewalking

Next, the command line is examined for a **treewalk pathname**, that is, a pathname whose directory part contains a wildcard. A **directory part** is the part before the final name. Each command can contain only one treewalk pathname. (More than one treewalk pathname may appear on the command line, provided each iteration yields no more than one.)

The command processor opens the directory whose pathname appears before the directory wildcard in the treewalk pathname. It visits the subdirectories of the tree and substitutes the pathname of each directory visited for the part of the treewalk pathname to the left of and including the directory wildcard. (See Chapter 4 of this book for details on treewalking.)

For example,

```
command A>B>@@>@.LIST
```

could execute the following series of commands:

```
command A>B>C>@.LIST
command A>B>C>X>@.LIST
command A>B>C>Y>@.LIST
command A>B>D>@.LIST
```

Commands of this series are passed on to the wildcarding step, below.

Matching Wildcard Names

Each command is scanned for a pathname whose last element (entryname) is a wildcard. Only one such pathname per command can be received at this step.

At the same time, the command is scanned for wildcard options (such as –AFTER) that specify selection criteria in addition to the wildcard name. These wildcard options are used only at this step in processing. Therefore, they are removed from the commands that are actually executed.

The command processor opens the directory given by the treename, and selects those entries in the directory that match the wildcard and the other selection criteria. If verify mode is enabled, the command processor asks the user to approve or disapprove each match.

The wildcard part of the pathname is then replaced with the actual name that is matched, and the command is passed to the next step.

For example,

```
command @.LIST -AFTER 12-1 -FILE
```

might execute as if it were the series

```
command A.LIST
command B.LIST
command C.LIST
    .
    .
    .
```

Matching Name Generation Patterns

The final step is name generation. The command processor searches for any pathname in the command that contains an equal sign (=) in the entryname. Any number of such pathnames is permitted. Each generation pattern (that is, each name

containing an equal sign) is replaced by the name it generates. The **source name** is usually the first object argument to the command, but individual commands may differ.

For example,

```
command ABC.LIST ==.+OLD
```

executes as

```
command ABC.LIST ABC.LIST.OLD
```

Executing the Command

The command that emerges from the name generation step is then executed. After execution (whether or not the command produced an error message), the next wildcard match, then the next treewalk step, and finally the next simple iteration step, is taken.

Example of Command-line Processing

The example below follows a command line, step-by-step, through processing to illustrate the order in which command-line processing occurs. The command listed after "CURRENT:" is the command currently being processed. The commands listed at "WAITING:" show, in order, subsequent commands to be executed. (The stack of waiting command lines shown in this example would not be created by PRIMOS. The commands are shown in this form for easier reading.)

The directory used in this example is shown in Figure 5-2.

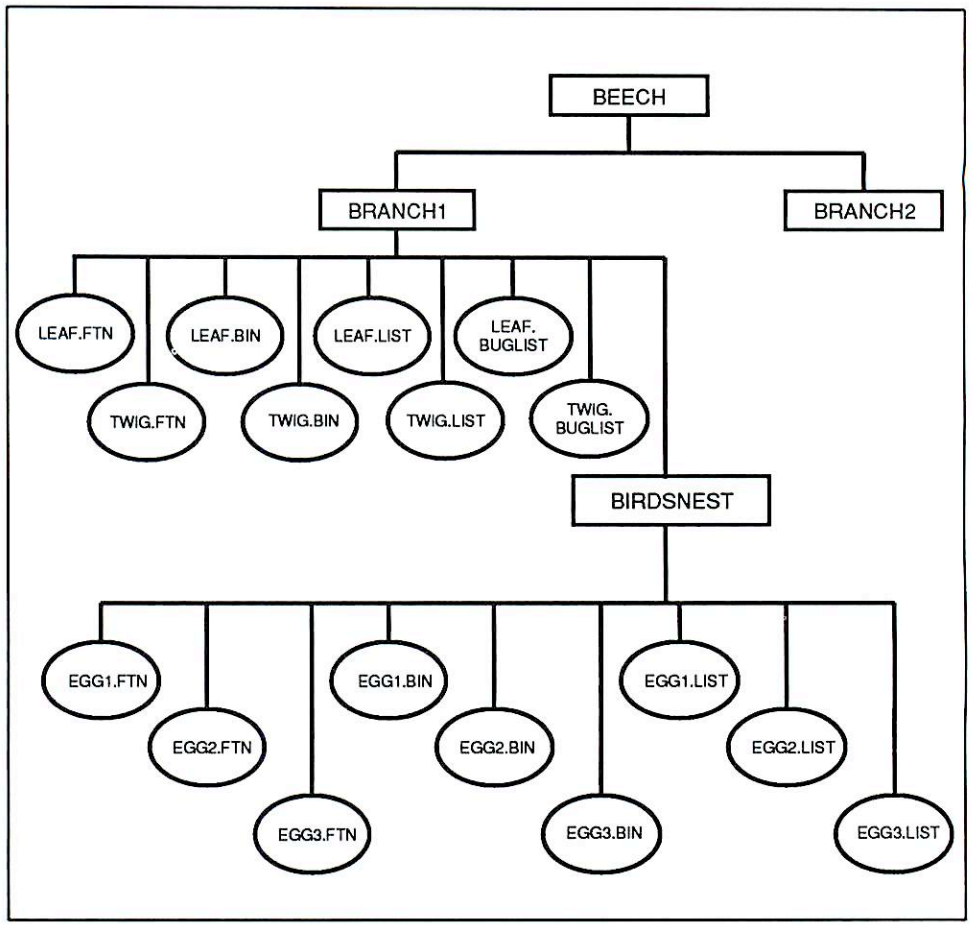


FIGURE 5-2
Sample Directory Tree

Processing Steps

1. The user types the following command line:

```
CHECK %.PROJ1%
```

2. Abbreviation Expansion: The command processor expands the user's abbreviation CHECK, producing the command line

```
SPOOL %.PROJ1%>(@.LIST @.BUGLIST); DELETE %.PROJ1%>(@.BIN @.LIST)
```

3. Syntax Suppression: The command line does not begin with a tilde (~). No syntax suppression occurs and processing continues.

4. Multiple Commands: The presence of a semicolon breaks the command line into two commands. The processing status is

```
CURRENT:    SPOOL %.PROJ1%>(@.LIST @.BUGLIST)
WAITING:    DELETE %.PROJ1%>(@.BIN @.LIST)
```

5. Variable Evaluation: The global variable %.PROJ1% is removed from the current command and replaced with its value BEECH>@@ from the global variable file previously activated. The processing status is

```
CURRENT:    SPOOL BEECH>@@>(@.LIST @.BUGLIST)
WAITING:    DELETE %.PROJ1%>(@.BIN @.LIST)
```

6. Command Function Evaluation: The command line contains no square brackets ([ ]), which denotes the absence of function calls. No change to the command occurs.

7. Iteration Evaluation: Next, the iteration lists are evaluated. The presence of an iteration list containing two items (@.LIST and @.BUGLIST); creates two command lines to replace the current command line. The first of the new command lines becomes the current line. The second is placed at the top of the waiting list. The processing status is

```
CURRENT:    SPOOL BEECH>@@>@.LIST
WAITING:    SPOOL BEECH>@@>@.BUGLIST
            DELETE %.PROJ1%>(@.BIN @.LIST)
```

8. Treewalking: Next, treewalking evaluation matches @@ against all directories in BEECH's subtree. Each match found creates a new command line. The first becomes current while the rest are placed at the top of the waiting list. The processing status is

```
CURRENT:    SPOOL BEECH>BRANCH1>@.LIST
WAITING:    SPOOL BEECH>BRANCH1>BIRDSNEST>@.LIST
            SPOOL BEECH>BRANCH2>@.LIST
            SPOOL BEECH>@@>@.BUGLIST
            DELETE %.PROJ1%>(@.BIN @.LIST)
```

9. Wildcard Matching: Matching of wildcards is done for the current command. The first match found replaces the wildcard in the current command line. Subsequent matches generate new commands for the waiting list. The processing status is

```
CURRENT:    SPOOL BEECH>BRANCH1>LEAF.LIST
WAITING:    SPOOL BEECH>BRANCH1>TWIG.LIST
            SPOOL BEECH>BRANCH1>BIRDSNEST>@.LIST
            SPOOL BEECH>BRANCH2>@.LIST
            SPOOL BEECH>@@>@.BUGLIST
            DELETE %.PROJ1%>(@.BIN @.LIST)
```

10. Name Generation: The current command is checked for name generation patterns. There are none, so no changes occur at this step.

11. Execution: The current command

```
SPOOL BEECH>BRANCH1>LEAF.LIST
```

is executed. The first command on the waiting list now becomes the current command. The processing status is

```
CURRENT:  SPOOL BEECH>BRANCH1>TWIG.LIST
WAITING:  SPOOL BEECH>BRANCH1>BIRDSNEST>@.LIST
          SPOOL BEECH>BRANCH2>@.LIST
          SPOOL BEECH>@@>@.BUGLIST
          DELETE %.PROJ1%>(@.BIN @.LIST)
```

12. Name Generation: The current command is checked for name generation patterns and then executed. The next command then moves up to become the current command. The processing status is

```
CURRENT:  SPOOL BEECH>BRANCH1>BIRDSNEST>@.LIST
WAITING:  SPOOL BEECH>BRANCH2>@.LIST
          SPOOL BEECH>@@>@.BUGLIST
          DELETE %.PROJ1%>(@.BIN @.LIST)
```

13. Wildcard Matching: Matching on the current command produces

```
CURRENT:  SPOOL BEECH>BRANCH1>BIRDSNEST>EGG1.LIST
WAITING:  SPOOL BEECH>BRANCH1>BIRDSNEST>EGG2.LIST
          SPOOL BEECH>BRANCH1>BIRDSNEST>EGG3.LIST
          SPOOL BEECH>BRANCH2>@.LIST
          SPOOL BEECH>@@>@.BUGLIST
          DELETE %.PROJ1%>(@.BIN @.LIST)
```

14. Name Generation: The current command, and then the next two, are checked for name generation. The current command is then executed, which results in the following processing status:

```
CURRENT:  SPOOL BEECH>BRANCH1>BIRDSNEST>EGG2.LIST
WAITING:  SPOOL BEECH>BRANCH1>BIRDSNEST>EGG3.LIST
          SPOOL BEECH>BRANCH2>@.LIST
          SPOOL BEECH>@@>@.BUGLIST
          DELETE %.PROJ1%>(@.BIN @.LIST)
```

15. Execution: The current command is executed, which produces the following status:

```
CURRENT:  SPOOL BEECH>BRANCH1>BIRDSNEST>EGG3.LIST
WAITING:  SPOOL BEECH>BRANCH2>@.LIST
          SPOOL BEECH>@@>@.BUGLIST
          DELETE %.PROJ1%>(@.BIN @.LIST)
```

16. Execution: The current command is executed, which produces the following status:

```
CURRENT:   SPOOL BEECH>BRANCH2>@.LIST
WAITING:   SPOOL BEECH>@@>@.BUGLIST
           DELETE %.PROJ1%>(@.BIN @.LIST)
```

17. Wildcard Matching: The current command requires wildcard matching. However, there are no wildcard matches for the directory BRANCH2. The current command line never executes, and the next command becomes current. The processing status is

```
CURRENT:   SPOOL BEECH>@@>@.BUGLIST
WAITING:   DELETE %.PROJ1%>(@.BIN @.LIST)
```

18. Treewalking: Treewalking evaluation produces the following status:

```
CURRENT:   SPOOL BEECH>BRANCH1>@.BUGLIST
WAITING:   SPOOL BEECH>BRANCH1>BIRDSNEST>@.BUGLIST
           SPOOL BEECH>BRANCH2>@.BUGLIST
           DELETE %.PROJ1%>(@.BIN @.LIST)
```

19. Wildcard Expansion: Wildcard expansion produces the following status:

```
CURRENT:   SPOOL BEECH>BRANCH1>LEAF.BUGLIST
WAITING:   SPOOL BEECH>BRANCH1>TWIG.BUGLIST
           SPOOL BEECH>BRANCH1>BIRDSNEST>@.BUGLIST
           SPOOL BEECH>BRANCH2>@.BUGLIST
           DELETE %.PROJ1%>(@.BIN @.LIST)
```

20. Name Generation: The top two commands are each checked for name generation. The first command is executed, resulting in the following processing status:

```
CURRENT:   SPOOL BEECH>BRANCH1>TWIG.BUGLIST
WAITING:   SPOOL BEECH>BRANCH1>BIRDSNEST>@.BUGLIST
           SPOOL BEECH>BRANCH2>@.BUGLIST
           DELETE %.PROJ1%>(@.BIN @.LIST)
```

21. Execution: The second command from step 20 is executed, producing the following status report:

```
CURRENT:   SPOOL BEECH>BRANCH1>BIRDSNEST>@.BUGLIST
WAITING:   SPOOL BEECH>BRANCH2>@.BUGLIST
           DELETE %.PROJ1%>(@.BIN @.LIST)
```

22. Wildcard Matching: Matching produces two more null matches. The top two commands do not execute, and the next command becomes current. Steps 23 to 43 result from this command:

```
CURRENT:   DELETE %.PROJ1%>(@.BIN @.LIST)
```

23. Variable Evaluation: The global variable %.PROJ1% is removed from the current command and replaced with its value BEECH>@@, which results in the command

```
CURRENT:   DELETE BEECH>@@>(@.BIN @.LIST)
```

24. Function Evaluation: No function calls are present and the processing status remains unchanged.

25. Iteration Evaluation: The presence of an iteration list containing two items (@.BIN and @.LIST) creates two command lines, the first of which becomes the current command. The processing status is

```
CURRENT:   DELETE BEECH>@@>@.BIN
WAITING:   DELETE BEECH>@@>@.LIST
```

26. Treewalking: Treewalking evaluation matches @@ against all directories in BEECH's subtree. Each match found creates a new command line. The processing status is

```
CURRENT:   DELETE BEECH>BRANCH1>@.BIN
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>@.BIN
           DELETE BEECH>BRANCH2>@.BIN
           DELETE BEECH>@@>@.LIST
```

27. Wildcard Matching: The result of matching wildcards for the current command is that two commands replace the current one. The processing status is

```
CURRENT:   DELETE BEECH>BRANCH1>LEAF.BIN
WAITING:   DELETE BEECH>BRANCH1>TWIG.BIN
           DELETE BEECH>BRANCH1>BIRDSNEST>@.BIN
           DELETE BEECH>BRANCH2>@.BIN
           DELETE BEECH>@@>@.LIST
```

28. Name Generation: There are no name generation patterns in the current command. The command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH1>TWIG.BIN
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>@.BIN
           DELETE BEECH>BRANCH2>@.BIN
           DELETE BEECH>@@>@.LIST
```

29. Execution: The current command is executed. The processing status is

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>@.BIN
WAITING:   DELETE BEECH>BRANCH2>@.BIN
           DELETE BEECH>@@>@.LIST
```

30. Wildcard Matching: Matching of wildcards replaces the wildcard name in the current command and generates new commands for the waiting list. The processing status is

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG1.BIN
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG2.BIN
           DELETE BEECH>BRANCH1>BIRDSNEST>EGG3.BIN
           DELETE BEECH>BRANCH2>@.BIN
           DELETE BEECH>@@>@.LIST
```

31. Execution: The current command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG2.BIN
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG3.BIN
           DELETE BEECH>BRANCH2>@.BIN
           DELETE BEECH>@@>@.LIST
```

32. Execution: The current command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG3.BIN
WAITING:   DELETE BEECH>BRANCH2>@.BIN
           DELETE BEECH>@@>@.LIST
```

33. Execution: The current command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH2>@.BIN
WAITING:   DELETE BEECH>@@>@.LIST
```

34. Wildcard Matching: Nothing matches the wildcard, so the DELETE command does not execute. The processing status is

```
CURRENT:   DELETE BEECH>@@>@.LIST
```

35. Treewalking: The last treewalk generates new command lines. The processing status is

```
CURRENT:   DELETE BEECH>BRANCH1>@.LIST
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>@.LIST
           DELETE BEECH>BRANCH2>@.LIST
```

36. Wildcard Matching: Matching of wildcard names replaces the wildcard name of the current command with an entryname. New commands are generated and placed on the waiting list. The processing status is

```
CURRENT:   DELETE BEECH>BRANCH1>LEAF.LIST
WAITING:   DELETE BEECH>BRANCH1>TWIG.LIST
           DELETE BEECH>BRANCH1>BIRDSNEST>@.LIST
           DELETE BEECH>BRANCH2>@.LIST
```

37. Execution: The current command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH1>TWIG.LIST
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>@.LIST
           DELETE BEECH>BRANCH2>@.LIST
```

38. Wildcard Matching: A found match in the current command replaces the wildcard symbol @:

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>@.LIST
WAITING:   DELETE BEECH>BRANCH2>@.LIST
```

39. Execution: The current command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG1.LIST
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG2.LIST
           DELETE BEECH>BRANCH1>BIRDSNEST>EGG3.LIST
           DELETE BEECH>BRANCH2>@.LIST
```

40. Execution: The current command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG2.LIST
WAITING:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG3.LIST
           DELETE BEECH>BRANCH2>@.LIST
```

41. Execution: The current command is executed, resulting in the following processing status:

```
CURRENT:   DELETE BEECH>BRANCH1>BIRDSNEST>EGG3.LIST
WAITING:   DELETE BEECH>BRANCH2>@.LIST
```

42. Wildcard Matching: The last wildcard matching is performed:

```
CURRENT:   DELETE BEECH>BRANCH2>@.LIST
```

43. Execution: Nothing matches the wildcard, so the DELETE command does not execute. The command line of step 1 has been completely processed and executed.

Terminal Display

When the command in the example executes, you see the response at the terminal shown in Figure 5-3.

```
OK, CHECK %.PROJ%
[SPOOL Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Request 11 added to queue, 1 records : <SYSONE>BEECH>BRANCH1>LEAF.LIST
[SPOOL Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Request 12 added to queue, 1 records : <SYSONE>BEECH>BRANCH1>TWIG.LIST
[SPOOL Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Request 13 added to queue, 1 records : <SYSONE>BEECH>BRANCH1>BIRDSNEST>EGG1.LIST
[SPOOL Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Request 14 added to queue, 1 records : <SYSONE>BEECH>BRANCH1>BIRDSNEST>EGG2.LIST
[SPOOL Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Request 15 added to queue, 1 records : <SYSONE>BEECH>BRANCH1>BIRDSNEST>EGG3.LIST
[SPOOL Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Request 16 added to queue, 1 records : <SYSONE>BEECH>BRANCH1>LEAF.BUGLIST
[SPOOL Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Request 17 added to queue, 1 records : <SYSONE>BEECH>BRANCH1>TWIG.BUGLIST
(std$cp) Verify wildcard selections for "BEECH>BRANCH1>@.BIN" :
"LEAF.BIN"? YES
"TWIG.BIN"? YES
(std$cp) Verify wildcard selections for "BEECH>BRANCH1>BIRDSNEST>@.BIN" :
"EGG1.BIN"? YES
"EGG2.BIN"? YES
"EGG3.BIN"? YES
(std$cp) Verify wildcard selections for "BEECH>BRANCH1>@.LIST":
"LEAF.LIST"? YES
"TWIG.LIST"? YES
(std$cp) Verify wildcard selections for "BEECH>BRANCH1>BIRDSNEST>@.LIST":
"EGG1.LIST"? YES
"EGG2.LIST"? YES
"EGG3.LIST"? YES
OK,
```

FIGURE 5-3
Sample Terminal Display

6 Command-line Editing

Introduction

EDIT_CMD_LINE (ECL) is a command-line editor that allows you to control command-line input to the terminal. On the simplest level, this means that you don't have to retype an entire command line if you've made a mistake. More importantly, ECL is a sophisticated way to manipulate command-line input. This chapter, divided into four sections, documents several important features of ECL.

- The section titled Introduction provides general information on ECL.
- The section titled Automatic Pathname Completion details how you can complete partially-entered pathnames, replace pathnames, and perform a treewalk wildcard expansion.
- The section titled ECL Key Bindings describes ECL's default key bindings, the key bindings file, key binding customization, and Programmable Function (PF) commands.
- The section titled ECL Commands Reference lists all of the ECL commands in alphabetical order and provides a description of each command.

For information about the ECL command and its command-line options, see Chapter 2. For additional information about ECL, including tutorial information, see the *PRIMOS User's Guide*.

Control Characters

Control characters entered into the command line are displayed by ECL as a two-character sequence: ^ (or ~ for the Prime Extended Character Set) followed by the corresponding ASCII-7 letter for the control character (for example, the ASCII mnemonic **DEL** is displayed as **^?**, and the Prime ECS mnemonic **SSA** is displayed

as ~F). Even though they appear as two characters on the screen, they are still treated as a single character internally. (See Appendix D for a complete listing of the Prime Extended Character Set.)

ECL Commands

EDIT_CMD_LINE must have a way to distinguish between those commands that you are giving to ECL and those commands that you are giving to some other part of the system.

All ECL commands start with either the Esc key or the Ctrl key. To issue a command using Esc, press it and then release it before you press the accompanying character key. To issue a command using Ctrl, press Ctrl and the character key *simultaneously*.

ECL commands can use a numeric argument count, **n**, either to repeat the same command **n** number of times, or to instruct ECL to use a different variation of the command. To repeat the same command **n** number of times, use the *esc_digit* command, described in the section Commands Reference at the end of this chapter. To use a different variation of the command, if applicable, use the *multiplier* command, also described in the section Commands Reference. These two commands are also discussed in the *PRIMOS User's Guide*.

The Kill Ring

Several commands use **kill buffers** that are themselves part of a **kill ring**. The kill ring is simply a circular ring of ten kill buffers. The most recent text deletion replaces the oldest text deletion in the kill ring. Consecutive text deletions are accumulated in the current kill buffer, rather than having each one take up a ring entry.

The Search Ring

The search commands *forward_search* and *reverse_search* maintain a separate search ring, or history of search strings. This ring is currently ten deep and is accessible from within a search prompt by using the *next_line*, *prev_line*, or *goto_line* commands, or by repeated searching through it.

Suspending the PRIMOS Break Character

The commands *do_echo*, *do_quote*, and *quote* all suspend the interpretation of the PRIMOS break character, Ctrl P, so that it can be echoed or quoted as a real Ctrl P. In order for this to work, the Ctrl P to be quoted or echoed cannot be entered using the typeahead feature of PRIMOS.

The suspension of the break character can lead to a confusing situation. For example, suppose you accidentally enter *do_echo* or *do_quote* and you don't know how to exit.

Typing Ctrl P does not allow you to break out because of the changed interpretation. In this case, type Ctrl P twice (or more) in rapid succession in order to break out of the process.

ECL as an EPF Command Function

When invoked as an EPF command function, ECL returns as its value a string representing the package version number followed by the internal data structure version number, separated by a space (for example, `303 302`). This is useful if, for example, the code in your CPL program references one or both of these numbers.

ECL and Command Output (COMO) Files

While a COMO file is open, ECL keeps the file clean by limiting output to the submitted command and its prompt. This aids the viewing, editing, and printing of such files. See also the –CLEAN_COMO / –NO_CLEAN_COMO options, documented in Chapter 2, for more information about controlling ECL terminal output to a COMO file.

Automatic Pathname Completion

ECL has a facility called automatic pathname completion. You type in part of the pathname, and ECL attempts to complete it for you. This is done with the *expand_wild* command. *expand_wild* attempts to complete the partially-typed pathname that the cursor is positioned within. The partially-typed pathname you enter must conform to the currently implemented PRIMOS pathname standards.

The *expand_wild* command permits you to specify a pathname either above or below your current position in the directory tree. Typing *< references the directory above the current one, *<< the directory two levels above, and so on. You can then reference downward from such a point. Thus, to reference another branch in the directory tree, type *<BRANCH2 (if you are one level below BRANCH2) or *<>BRANCH2 (if you are in a different branch of the directory tree).

How ECL Completes Pathnames

The following procedure describes the method ECL uses to complete partial pathnames with the *expand_wild* command.

1. The partially-typed pathname is extracted from the command line.
2. ECL checks for a tilde (~) in Column 1 of the command line, indicating suppression of abbreviation expansion.
3. If the tilde is not present, and if ECL's –WILD_ABBREV option is enabled, all pathname entrynames *except* the last (or the entryname the cursor is within when the –WILD_TAIL option is in effect) are passed through the PRIMOS abbreviation preprocessor for expansion of argument-type abbreviations and global variable references.

4. The PRIMOS wildcard characters **@@** are appended to the command line (or they are inserted at the cursor position within the pathname if the –WILD_TAIL option is in effect).
5. A wildcard expansion routine processes the command line. This produces for ECL a list of full pathnames that correspond to all existing PRIMOS file system objects matching the wildcard pattern. (Pathnames containing embedded passwords are not supported.)
6. If the list is empty, the terminal bell is rung, indicating that no file system objects fulfilled the completion match criteria.
7. If the list contains only one entry, that entire entry is inserted, and the pathname is completed. The completed pathname automatically includes a final > if the file system object is a directory, although the –NO_WILD_DIRECTORY option overrides this behavior.
8. If more than one match is found, all the entries in the list are scanned to determine the common initial substring that identifies the set unambiguously.
9. The substring is inserted into the command line, replacing the unexpanded form, and the cursor is placed at the end of this insertion. If the –WILD_TAIL option was specified, the common tail substring is also determined and inserted after the cursor. The pathname you have typed is expanded as much as possible, and the cursor position indicates the point of ambiguity.
10. If the command line pathname cannot be completed further without ambiguity, or if the numeric argument to the *expand_wild* command is 4, then the list of matching entries is displayed on the terminal as a **menu**. When displayed, directories have a > appended to their names for identification purposes. The list is alphabetically sorted and numbered across columns (by default) or numbered down rows if the –COL_MAJOR option is in effect. If the list is lengthy, the display pauses after every screenful. A space bar or carriage return at the **--More--** prompt displays the next screenful; **q**, **n**, **Q**, or **N**, abort the display; any other character displays the next line and reprompts.
11. At this point, you can choose to enter more of the pathname for completion or select the desired entry from the menu list using the *expand_wild_menu* command. If you prefer the incremental completion method instead of the menu, you can turn off the menu-list numbering feature with the –NO_WILD_MENU option.

Replacing Pathnames

You can replace the pathname you entered with its equivalent full PRIMOS pathname within your command line. To do this, specify an **n** count of 16 for the *expand_wild* command (*multiplier multiplier expand_wild*). This also causes abbreviation expansion of *all* pathname entrynames, including the last one (or the one that the cursor is within) when the –WILD_ABBREV option is in effect.

Treewalk Wildcard Expansion

When you specify the –WILD_TAIL option, you can perform a *treewalk* wildcard expansion. Do this by positioning the cursor before the final entryname of the partially-completed pathname (that is, before the last >).

ECL attempts to locate the matching directories (and subdirectories, subsubdirectories, and so forth) to complete the pathname. When displaying the list of matches, full pathnames are shown and the list is *not* sorted. Be aware that PRIMOS treewalk pathnames implicitly bind the treewalk wildcard expression to the end of the pathname rather than to the beginning. That is, additional directories can be inserted to the left of the first wildcarded entryname position. For example, if the following four files exist,

```
<DISK>USER>DIR1>SOMEFILE
<DISK>USER>XXX1>XXX2>DIR3>SOMEFILE
<DISK>USER>DIR1>XXX2>SOMEFILE
<DISK>USER>XXX1>XXX2>XXX3>SOMEFILE
```

then the treewalk pathname <DISK>USER>D@>SOMEFILE finds only the first two, while <DISK>USER>X@>SOMEFILE finds only the last two.

Example Session

The following is a sample dialog of a PRIMOS session. Commentary is provided between examples. The cursor position is shown using the underscore character. Completion requests are made with the *expand_wild* command.

```
OK, LD DISK1>USER1>DIRECTORY1>@@

<DISK1>USER1>DIRECTORY1 (ALL access)
20 records in this directory, 324 total records out of quota of 0.

3 Files.

FILE1.TXT          FILE2.TXT          ZERO

3 Directories.

PRODUCTS           SUBDIR1            SUBDIR2

OK, LD <DISK1>USER1>DIRECTORY1>F_
```

Requesting completion produces

```
OK, LD DISK1>USER1>DIRECTORY1>FILE_
```

The partial pathname <DISK1>USER1>DIRECTORY1>F can be completed up through the E in FILE, but no further because of the ambiguity with FILE1.TXT and FILE2.TXT. Requesting completion again by typing Ctrl I displays the list of possible completions because further completion is not possible:

```
1) FILE1.TXT    2) FILE2.TXT
```

Now continue by typing a 1 for the first entry on the menu:

```
OK, LD <DISK1>USER1>DIRECTORY1>FILE1_
```

Requesting completion by again typing Ctrl I at this point fully completes the pathname because FILE1 identifies a unique file in the indicated directory:

```
OK, LD <DISK1>USER1>DIRECTORY1>FILE1.TXT_
```

Requesting completion again displays the single matching entry:

```
1) FILE1.TXT
```

If you are not familiar with the directory contents but still want to attach to one of its subdirectories, do the following:

```
OK, ATTACH <DISK1>USER1>DIRECTORY1>_
```

At this point, there is no unambiguous expansion, so ECL lists the contents of the directory in a menu format.

```
1) FILE1.TXT    2) FILE2.TXT    3) PRODUCTS>    4) SUBDIR1>
5) SUBDIR2>     6) ZERO
```

The directories have the > character appended for easy identification. Now attach to the directory PRODUCTS by typing P and again requesting completion:

```
OK, ATTACH <DISK1>USER1>DIRECTORY1>P_
```

Here is the result:

```
OK, ATTACH <DISK1>USER1>DIRECTORY1>PRODUCTS>_
```

Note that the > has appeared because PRODUCTS is a directory. Alternatively, you can type Esc 3 Esc Ctrl I, invoking the *expand_wild_menu* command with a numeric argument of 3 to directly choose PRODUCTS from the menu list display.

Because the ATTACH command does not take the > as the last token on the command line, you must delete it before submission. Having done this, attach to PRODUCTS. Now, try to reference upward from the current directory.

```
OK, LD *<_
```

There is no unambiguous expansion in the parent directory, so ECL lists its contents:

```
1) FILE1.TXT    2) FILE2.TXT    3) PRODUCTS>    4) SUBDIR1>
5) SUBDIR2>     6) ZERO
```

To list the file ZERO, you need only to type

```
OK, LD *<Z_
```

and request completion to produce

```
OK, LD <DISK1>USER1>DIRECTORY1>ZERO_
```

The *< expression has been expanded automatically into a legal PRIMOS pathname. Finally, suppose that you want to create another directory in the directory above:

```
OK, CREATE *NEWDIR_
```

This produces an error if submitted because CREATE does not understand the new pathname syntax. Instead, request completion with a count of 16 (for example, by issuing the *multiplier multiplier* command), which replaces the pathname with its legal equivalent:

```
OK, CREATE <DISK1>USER1>DIRECTORY1>NEWDIR_
```

and finally submits this as your command line.

Now use the –WILD_TAIL option, and repeat the first expansion above:

```
OK, ECL -WILD_TAIL
OK, LD <DISK1>USER1>DIRECTORY1>F_
```

Requesting completion now produces

```
OK, LD <DISK1>USER1>DIRECTORY1>FILE_.TXT
```

where the cursor is under the period (.) of FILE.TXT. If the –WILD_TAIL option had not been used, .TXT would not have been displayed. Since .TXT is displayed, the cursor shows where further completion is necessary. Requesting completion again shows the menu-list of possible matches:

```
1) FILE1.TXT   2) FILE2.TXT
```

To choose FILE2.TXT, simply type 2; the cursor is already where it should be to insert this:

```
OK, LD <DISK1>USER1>DIRECTORY1>FILE2_.TXT
```

Requesting completion here moves the cursor to the end, which indicates that it is complete:

```
OK, LD <DISK1>USER1>DIRECTORY1>FILE2.TXT_
```

Note that with –WILD_TAIL, you can complete an entryname even if you don't know how it begins. Suppose you want a file ending in TXT:

```
OK, OPEN <DISK1>USER1>DIRECTORY1>TXT
```

By placing the cursor under the T of TXT and requesting completion, you get exactly what you got previously when you expanded <DISK1>USER1>DIRECTORY1>F because only FILE1.TXT and FILE2.TXT end with TXT:

```
OK, LD <DISK1>USER1>DIRECTORY1>FILE_.TXT
```

The cursor is under the . of FILE.TXT.

ECL Key Bindings

Each key on your terminal is *bound to* a specific ECL command. In other words, when you press a key you execute the command that is associated with, or bound to, that key. For example, within ECL [Ctrl] [A] is bound to the *begin_line* command, and issuing the command [Ctrl] [A] positions the cursor at the beginning of the current line. Most of the keys on the main keypad are bound to the *self_insert* command; when you press one of them, the character that appears on that key is echoed on the terminal screen.

You can change these default bindings to suit your needs by binding any character or sequence of characters to any ECL command. To define specific functions for specific keys, you must perform the following steps:

1. Create a **bindings file** with a standard text editor. A bindings file is a text file containing one binding per line. This file may have any legal PRIMOS filename.
2. Use the ECL –BIND_TERM option to compile and load the bindings.

To save time and increase efficiency, it is recommended that you also perform these additional steps:

1. Use the ECL –SAVE_TERM option to save the compiled bindings in a separate file.
2. Use the ECL –RESTORE_TERM option to quickly reload this file during a subsequent session.

The –SAVE_TERM and –RESTORE_TERM options are discussed later in the section titled Compiling, Saving, and Restoring the Bindings File.

Syntax of the Bindings File

The syntax and syntax descriptions of the bindings file are given below. Each binding must be on a separate line, with blank spaces or tab settings between the fields but not within any of them. (You can include comments in the file by entering the number character (#) in the first column of a line.)

sequence command[|program] [terminal_list]

The sequence Argument: **sequence** is the character or character sequence that you want to bind to a particular ECL command. The **sequence** field can contain the character sequence sent by any key on your terminal. Two major categories of characters are printing characters and nonprinting characters.

Printing characters

Most of the keys on the main keypad send printing characters because when you press one of them the character that appears on the keypad is echoed on the screen. You can specify printing characters as part of **sequence** simply by entering them in the bindings file.

Nonprinting characters

Several keys, or key combinations such as Ctrl A, do not send a printing character to be echoed on the terminal screen. Instead, a nonprinting character or sequence of nonprinting characters is sent. Keys such as Backspace, Tab, and Esc send nonprinting characters. Function keys such as F1 through F10, PF1 through PF12, and PA1 through PA4 (depending on your terminal) send a sequence of nonprinting (and printing) characters. You cannot specify nonprinting characters by their keypad name, as you can with printing characters. Instead, you must use the methods discussed below in the section Other Nonprinting Character Sequences.

The command Argument: **command** is the name of the ECL command to be invoked by the character or character sequence (for example, *begin_line*). Refer to the section ECL Commands Reference for a complete list of command names.

The program Argument: **program** is the function to be programmed to an ECL programmable function command. If you use the **program** argument, **command** either must be one of ECL's programmable function commands (pfkey1 to pfkey512), or it must be *execute_macro* to specify the initial keyboard macro program. Also, you must include the vertical bar character (|) between *command* and *program*. Spaces are not permitted.

The terminal_list Argument: **terminal_list** specifies the terminal type(s) that apply to this binding. If a particular binding has more than one terminal type, separate them by the vertical bar character (|) with no spaces in between. If you do not include this argument, ECL performs the binding regardless of what kind of terminal you are using. You also can enter the vertical bar character, either alone or at the beginning or end of the **terminal_list**, to include ECL's default generic 24-line by 80-column terminal in the list of terminals for the binding.

Note

The number character (#) cannot appear in Column 1 in your bindings file because ECL interprets this character as a comment character in that position. To use the number character as a character other than a comment character, leave a space in Column 1.

Key Binding Sequences

You can specify character keys for the ECL binding sequences in two ways.

Control Sequences With Printing Characters: Use the caret (^) to indicate a control sequence. For example, the representation ^A in a bindings file specifies Ctrl A.

Other Nonprinting Character Sequences: The backslash character (\) is the first character in the representation of any nonprinting character that is part of **sequence**. You can also use the backslash to specify printing characters, such as the caret (represented by \^) and the backslash character itself (represented by \\). Several backslash character combinations have special meanings, as shown in the following list.

Sequence	*Meaning*
\b	The BACKSPACE key (same as ^H)
\E	The ESC key (same as ^[ )
\f	A form-feed character (same as ^L)
\l	ASCII linefeed character
\n	A linefeed character
\r	ASCII carriage return character (same as ^M)
\t	The TAB key (same as ^I)
\v	The vertical tab character (same as ^K)

Use the backslash character to include the octal representation of a character in the sequence field. For instance, you can use \205 in the bindings file to denote Ctrl E. Also, if you must include a space in the **sequence** or **program** argument, you must use the representation \240 (the octal representation of a space character).

Preceding any other printing character with the backslash results in binding only that character to the ECL command specified. For example, if you bind \x to *prev_line*, ECL displays the previous command line when you press x.

As an example, suppose the bindings file for a PT45 contains these two lines:

```
\EC      forward_char      pt45|dm30
\ED      back_char         pt45|dm30
```

The first line shown above sets the sequence *\EC* (Esc C) as the command to move forward one character (*forward_char*) on a PT45 terminal. (The right arrow key → is the same as Esc C on a PT45.)

Similarly, the next line shows that the sequence *\ED* (Esc D) is the key binding for the command to move backward one character (*back_char*). (The left arrow key ← is the same as Esc D on a PT45.) Be aware that the sequence field in the bindings file is case-sensitive, but the command field and the terminal field are not.

Table 6-1 lists each character of the ASCII subset of the Prime ECS character set. This table contains character codes most often used in a bindings file. The left side of each column contains the ASCII mnemonic, and the right side of each column shows the corresponding character sequence. For example, the BS mnemonic is represented in the bindings file as ^H (Ctrl H). Refer to Appendix D for the complete Prime ECS character set.

TABLE 6-1
Quick-reference ASCII Character Chart

NUL	^@	DLE	^P	SP	\240	0	0	@	@	P	P	‘	‘	p	p
SOH	^A	DC1	^Q	!	!	1	1	A	A	Q	Q	a	a	q	q
STX	^B	DC2	^R	"	"	2	2	B	B	R	R	b	b	r	r
ETX	^C	DC3	^S	#	#	3	3	C	C	S	S	c	c	s	s
EOT	^D	DC4	^T	$	$	4	4	D	D	T	T	d	d	t	t
ENQ	^E	NAK	^U	%	%	5	5	E	E	U	U	e	e	u	u
ACK	^F	SYN	^V	&	&	6	6	F	F	V	V	f	f	v	v
BEL	^G	ETB	^W	’	’	7	7	G	G	W	W	g	g	w	w
BS	^H	CAN	^X	(	(	8	8	H	H	X	X	h	h	x	x
HT	^I	EM	^Y	)	)	9	9	I	I	Y	Y	i	i	y	y
LF	^J	SUB	^Z	*	*	:	:	J	J	Z	Z	j	j	z	z
VT	^K	ESC	\E	+	+	;	;	K	K	[	[	k	k	{	{
FF	^L	FS	^\	,	,	<	<	L	L	\	\\	l	l	|	|
CR	^M	GS	^]	-	-	=	=	M	M	]	]	m	m	}	}
SO	^N	RS	^^	.	.	>	>	N	N	^	\^	n	n	~	~
SI	^O	US	^_	/	/	?	?	O	O	_	_	o	o	DEL	^?

Commands Useful for ECL Key Bindings

This section introduces some fundamental ECL commands that are useful in setting your own bindings. These commands are discussed further in the sections that follow, and are also included in the complete list of ECL commands at the end of the chapter.

do_echo Esc @

Echoes to the terminal the characters that follow without interpreting them as being part of a command. Echoing discontinues when *do_echo* is repeated. For example, Esc @PRT Esc @ sends the string "PRT" to the terminal.

do_quote Esc '

Self-inserts the characters that follow directly into the command line without further processing. Quoting discontinues when *do_quote* is repeated. (This is useful for determining what a key on the terminal actually sends.)

echo_raw *not bound*

Echoes the bound character sequence at the terminal without including it in the command line. This is useful for keys such as Scroll Up/Down on a PST 100 so that you can scroll the screen locally without affecting the command line being entered.

explain_key ^ _

Prompts for the key sequence for which you need information and displays the command name bound to that sequence. This command is never collected within a keyboard macro. The numeric argument is ignored but saved for the next command entered.

quote Ctrl Q **or** Esc Q **or** Esc q

Quotes the next character that is typed for self-insertion. This permits the insertion of characters into the command line that would normally be bound to commands other than *self_insert*.

self_insert *printable characters*

Inserts the last character of the bound sequence into the command line at the current position.

submit Ctrl J

Submits the entire command line for processing. The cursor can be anywhere on the line. *submit* finishes a keyboard macro if the macro was begun at the current level; this makes the macro self-terminating.

unbound **[anything not otherwise bound]**

Disables any binding for this sequence. Everything not explicitly bound to another sequence is unbound. This is useful in nullifying a key whose effects are potentially dangerous.

Programmable Function Commands

Bindings can be made to Programmable Function commands (PF commands), and these bindings can be programmed in the **program** field in the bindings file. Binding ECL PF commands to the extra function keys on a terminal can be useful because they are not keys that you normally use to enter command line text. Also, you can reconfigure an undesired keyboard layout using PF command bindings.

You can use a maximum of 512 different PF commands in a bindings file. Each PF command value in **program** cannot exceed 200 characters, and the total number of characters you can use for programming all PF commands cannot exceed 8000 characters.

Note

PF commands are not necessarily synonymous to the special function keys on a terminal; PF commands can be any key or sequence of characters. In using PF commands, you must include in the **command** field one of the PF commands (pfkey1 through pfkey512) or *execute_macro* to specify the initial keyboard macro program.

How the PF Commands Work

In the bindings file, the program field (**program**) is the sequence of characters executed by the PF command. When a PF command is used, **program** is substituted for **sequence** and reinterpreted by ECL. For example, if the bindings file for a PT200 contains this line for f1:

```
\EO!            pfkey1          pt200
```

whenever you press the PT200's [f1] key, its default program is inserted into the command line. Now suppose you want to bind the PRIMOS LD command to this key. Change this line in the bindings file, following the syntax rules described earlier, so that the line now looks like this:

```
\EO!            pfkey1|LD           pt200
```

Next, recompile the bindings file:

```
OK, ECL -BIND_TERM ECL_BINDINGS -TTP PT200
Established PT200 terminal capabilities (9600 baud). (EDIT_CMD_LINE)
Established PT200 bindings from 'ECL_BINDINGS' (91% available). (EDIT_CMD_LINE)
OK,
```

For the duration of this terminal session, pressing [f1] executes the LD command. To keep this change in compiled form for future sessions, use the –SAVE_TERM and –RESTORE_TERM options.

Multiple Command Sequences

You can bind more than one **sequence** to the same PF command by programming the first binding to a particular PF command, and then making other normal (unprogrammed) bindings to the same PF command. If a normal binding is made to an unprogrammed PF command, this programs the key with a default of FUNC#, where # is the number (1 through 512) of the PF command.

Continuing the example in the previous section (when you programmed the LD command to [F1] on a PT200 terminal), suppose you want [F2] to execute the LD command just as [F1] does. You know that the sequence \EO" is bound by default to the PF command pfkey2 ([F2] on a PT200); here is its binding in the bindings file:

```
\EO"            pfkey2              pt200
```

To make [F2] execute the LD command as [F1] does, you don't have to put LD in the **program** field (pfkey2|LD) as you would expect. Instead, just change the PF command of [F2] to match the PF command of [F1], which executes the LD command. Thus, your bindings file looks like this:

```
\EO!            pfkey1|LD           pt200
\EO"            pfkey1              pt200
```

Now, both [F1] and [F2] execute LD.

Note

Be aware that there is no default keyboard macro program for the *execute_macro* command.

Self-terminating Function Keys

Normally, when you press a terminal's function key, a key-specific character sequence plus a termination sequence is transmitted. In other words, the character sequence associated with that function key is present along with an instruction to *submit* that character sequence. You could equate this to executing the LD command at PRIMOS level; you type the keys [L] [D] and then press a key, [Return], that instructs PRIMOS to submit, or terminate, the LD command. [Return] is bound to the ASCII sequence [Ctrl] [J] (represented in the bindings file as ^J) by default.

Function keys also have default termination sequences, but you can change this default if you so desire. For example, on the PT45, two switches at the back of the terminal allow you to terminate a function key with ^C, ^D, or ^J. On a PST 100 or a PT200, you can use software to change the default, and it can be set to nothing (which is the terminal default) or set to ^J. Since ^J is usually bound to the *submit* command, you can make function key bindings that self-terminate (submit themselves).

Suppose you have a PT45 terminal that terminates function keys with ^J. Now examine the contents of the following bindings file:

```
^B\Ep          pfkey1              pt45|dm30
^B\Eq          pfkey2|F2           pt45|dm30
^B\Er^J        pfkey3|\E@\ED\E@    pt45|dm30
^B\Es^J        pfkey4|\E'COPY\E'^J  pt45|dm30
```

1. Binding #1 binds the terminal's [f1] to the default programmed sequence **FUNC1** because no **program** is supplied. This binding self-terminates because ^J has been left off the bound **sequence**.
2. Binding #2 binds [f2] so that the string "F2" is inserted into the command line. (The characters "F" and "2" are bound to *self_insert*.) This function key also self-terminates.
3. Binding #3 makes [f3] work this way: echo, Escape-D, stop echo. (\E@ is bound to *do_echo*.) This causes Escape-D to be sent to the terminal. Notice that the ^J is added to the **sequence** field so that ^J is explicitly absorbed as part of the binding and *not* left to be interpreted as the *submit* command, since you don't want to submit Escape-D. This key, therefore, does not self-terminate.
4. Binding #4 binds [f4] to do the following: quote, "COPY", stop quote, terminate. (\E' is bound to *do_quote*.) Notice that ^J is in the **sequence** field to be considered as part of the binding sequence. The ^J in the **program** field causes COPY to be inserted and then submitted explicitly (self-terminated).

You can append ^J to the end of the **program** field in order to explicitly establish a self-terminating PF command. This is useful for terminals that cannot be set up to terminate function keys with ^J themselves.

Note

To avoid programming a PF command to another PF command, use the sequence bound to *quote* or *do_quote*. For example, suppose you don't like your keyboard layout and want to remap the caret (^) character and the tilde (~) character on a PT45 so that they self-insert one another. In this case, the following entries in the bindings file are illegal:

```
~          pfkey511|\^          pt45|dm30
\^         pfkey512|~           pt45|dm30
```

Because each character is now a PF command binding and no longer bound to *self_insert*, an endless loop results. The proper way to have these characters self-insert one another is to use the quote command (^Q), as follows:

```
~          pfkey511|^Q\^        pt45|dm30
\^         pfkey512|^Q~         pt45|dm30
```

Sample Bindings File

The sample bindings file shows how to represent various terminal keys. The file includes comments, denoted by # in the first position, that explain the key binding directly below. This file is for a PST 100 terminal, and all command descriptions assume a count of 1.

```
#############################################################################################
###
###
### PST100/PT200 bindings:
###
#############################################################################################
#
# Sequence                Command|Program              Terminal
# --------                ---------------              --------
#
# pressing Esc n moves the cursor to the next line
# (\e = Escape key)
#
\en                       next_line                    pst100
#
# pressing Esc p moves the cursor to the previous line
#
#
\ep                       prev_line                    pst100
#
# pressing Ctrl a or Ctrl A moves the cursor one word to the right
# (^ = Control key)
#
^a                        forward_word                 pst100
```

```
#
# pressing the caret character and the letter a moves the cursor
# one character to the right
# (\^ = caret key)
#
\^a                         forward_char                    pst100
#
# pressing Ctrl B (which is 202 octal) moves the cursor to the
# beginning of the line
#
\202                        begin_line                      pst100
#
#
```

Bindings for Terminals

Here is the standard bindings file for the PT45, PST 100, and PT200 terminals. In the listing shown here, the designation pt200 actually represents the four different kinds of PT200: the standard PT200, the PT200 in wide 132-column mode, the color PT200, and the color PT200 in wide 132-column mode.

```
#############################################################################################
###
### PST100/PT200 bindings:
###
#############################################################################################
#
# Sequence Command        Terminal      <Key>

# -------- -------        --------      ------------------------------------

#
# ^H       rubout_char    pst100         # <BACKSP>-ALL  (PST100)
# ^H       rubout_char           pt200   # <BACKSP>-U/S   (PT200)
# ^I       expand_wild    pst100         # <TAB>-ALL     (PST100)
# ^I       expand_wild           pt200   # <TAB>-U/S      (PT200)
# ^J       submit         pst100|pt200   # <RETURN>-ALL
# ^J       submit         pst100|pt200   # <ENTER>-U/S
# ^N       echo_raw       pst100         # <CHAR SET>-U/S(PST100)
# ^N       echo_raw              pt200   # <CHAR SET>-ALL (PT200)
# ^O       echo_raw       pst100         # <CHAR SET>-U/S(PST100)
# ^O       echo_raw              pt200   # <CHAR SET>-ALL (PT200)
# ^P       unbound        pst100         # <STOP>-U      (PST100)
# ^P       unbound        pst100         # <CONFIG>-C    (PST100)
# \E       (Prefix)       pst100|pt200   # <ESC>-ALL
  \E$A     echo_raw       pst100|pt200   # <HOME>-U                Cursor Relative Home
  \E$B     echo_raw       pst100|pt200   # <HOME>-S/C/CS           Cursor Absolute Home
  \E$G     echo_raw       pst100         # <RESET>-U     (PST100) Reset
  \E$G     echo_raw              pt200   # <CLEAR>-U      (PT200) Reset
  \E$a     echo_raw              pt200   # <PREV>-U       (PT200) Page Up    (aka <PG UP>)
  \E$b     echo_raw              pt200   # <NEXT>-U       (PT200) Page Down (aka <PG DN>)
  \E?      echo_raw       pst100         # <RESET>-C     (PST100) Clear Screen
  \E?      echo_raw              pt200   # <CLEAR>-C      (PT200) Clear Screen
```

```
\E[0i     echo_raw       pst100          # <AUX SND>-U    (PST100) Media Copy Screen Transfer
\E[0i     echo_raw              pt200    # <PRT SCN>-U     (PT200) Media Copy Screen Print
\E[4h     unbound        pst100|pt200    # <INSERT>-ALL            Insert Mode On
\E[4l     unbound        pst100|pt200    # <INSERT>-ALL            Insert Mode Off
\E[>0i    echo_raw       pst100|pt200    # <AUX SND>-S    (PST100) Media Copy Area Transfer
\E[>1i    echo_raw       pst100          # <AUX SND>-C    (PST100) Media Copy Screen Transfer
\E[>1i    echo_raw              pt200    # <PRT SCN>-C     (PT200) Media Copy Screen Transfer
\E[>2i    echo_raw              pt200    # <PTR SCN>-S     (PT200) Media Copy Page Dump
\E[>10h   echo_raw              pt200    # <NUM LOCK>-ALL  (PT200) Function Keypad
\E[>10l   echo_raw              pt200    # <NUM LOCK>-ALL  (PT200) Numeric Keypad
\E[>20h   unbound        pst100          # <CHAR SET>-C   (PST100) DSC Mode Set
\E[>20l   unbound        pst100          # <CHAR SET>-CS  (PST100) DSC Mode Reset
\E[A      prev_line      pst100          # <UP>-ALL       (PST100) Cursor Up
\E[A      prev_line             pt200    # <UP>-U          (PT200) Cursor Up
\E[B      next_line      pst100          # <DOWN>-ALL     (PST100) Cursor Down
\E[B      next_line             pt200    # <DOWN>-U        (PT200) Cursor Down
\E[C      forward_char   pst100          # <RIGHT>-ALL    (PST100) Cursor Right
\E[C      forward_char          pt200    # <RIGHT>-U       (PT200) Cursor Right
\E[D      back_char      pst100          # <LEFT>-ALL     (PST100) Cursor Left
\E[D      back_char             pt200    # <LEFT>-U        (PT200) Cursor Left
\E[J      kill_line      pst100          # <ERASE>-C/CS   (PST100) Erase in Display to EOS
\E[J      kill_line             pt200    # <ERASE>-CS      (PT200) Erase in Display to EOS
\E[K      kill_line      pst100|pt200    # <ERASE>-S               Erase in Line to EOL
\E[M      echo_raw       pst100|pt200    # <DELETE>-S              Delete Line
\E[O      echo_raw       pst100|pt200    # <ERASE>-C       (PT200) Erase in Area to EOA
\E[P      delete_char    pst100|pt200    # <DELETE>-U              Delete Character
\E[S      echo_raw       pst100          # <SCROLL>-S     (PST100) Scroll Up
\E[S      echo_raw              pt200    # <SCRL UP>-U     (PT200) Scroll Up
\E[T      echo_raw       pst100          # <SCROLL>-U     (PST100) Scroll Down
\E[T      echo_raw              pt200    # <SCRL DN>-U     (PT200) Scroll Down
\E[1U     echo_raw       pst100          # <SCROLL>-C     (PST100) Next Page
\E[U      echo_raw              pt200    # <END/BEGIN>-U   (PT200) Next Page
\E[1V     echo_raw       pst100          # <SCROLL>-CS    (PST100) Preceding Page
\E[V      echo_raw              pt200    # <END/BEGIN>-S   (PT200) Preceding Page
\E[X      echo_raw       pst100|pt200    # <ERASE>-U               Erase Character
\E[Z      unbound        pst100          # <B-TAB>-ALL    (PST100) Cursor Backward Tabulation
\E[Z      unbound               pt200    # <B-TAB>-U/S     (PT200) Cursor Backward Tabulation
\E[j      echo_raw              pt200    # <SCRL RIGHT>-U  (PT200) Scroll Right
\E[k      echo_raw              pt200    # <SCRL LEFT>-U   (PT200) Scroll Left
\E]0\E\\  echo_raw       pst100|pt200    #                         Operating System Command
\E]1\E\\  echo_raw       pst100|pt200    #                         Operating System Command
\E]2\E\\  echo_raw       pst100|pt200    #                         Operating System Command
\E_0\E\\  echo_raw       pst100|pt200    # <MENU>-U                Application Program Command
\E_1\E\\  echo_raw       pst100|pt200    # <HELP>-U                Application Program Command
\E_2\E\\  echo_raw       pst100|pt200    # <HELP>-S                Application Program Command
\E_3\E\\  echo_raw       pst100          # <HELP>-C/CS    (PST100) Application Program Command
\E_3\E\\  echo_raw              pt200    # <HELP>-C        (PT200) Application Program Command
\E_4\E\\  echo_raw              pt200    # <HELP>-CS       (PT200) Application Program Command
\E_5\E\\  echo_raw              pt200    # <MENU>-S        (PT200) Application Program Command
\E_6\E\\  echo_raw              pt200    # <MENU>-C        (PT200) Application Program Command
\E_7\E\\  echo_raw              pt200    # <MENU>-CS       (PT200) Application Program Command
\EN!      unbound               pt200    # <PA1>-U         (PT200)
\EN%      unbound               pt200    # <PA1>-S         (PT200)
\EN)      unbound               pt200    # <PA1>-C         (PT200)
\EN-      unbound               pt200    # <PA1>-CS        (PT200)
\EN"      unbound               pt200    # <PA2>-U         (PT200)
```

```
\EN&      unbound          pt200   # <PA2>-S          (PT200)
\EN*      unbound          pt200   # <PA2>-C          (PT200)
\EN.      unbound          pt200   # <PA2>-CS         (PT200)
\EN#      unbound          pt200   # <PA3>-U          (PT200)
\EN'      unbound          pt200   # <PA3>-S          (PT200)
\EN+      unbound          pt200   # <PA3>-C          (PT200)
\EN/      unbound          pt200   # <PA3>-CS         (PT200)
\EN$      unbound          pt200   # <PA4>-U          (PT200)
\EN(      unbound          pt200   # <PA4>-S          (PT200)
\EN,      unbound          pt200   # <PA4>-C          (PT200)
\EN0      unbound          pt200   # <PA4>-CS         (PT200)
\EN1      unbound          pt200   # <SCRL LOCK>-U    (PT200)
\EN2      unbound          pt200   # <SCRL LOCK>-S    (PT200)
\EN3      unbound          pt200   # <SCRL LOCK>-C    (PT200)
\EN4      unbound          pt200   # <SCRL LOCK>-CS   (PT200)
\EN5      unbound          pt200   # <SM UNLABEL>-U   (PT200)
\EN6      unbound          pt200   # <SM UNLABEL>-S   (PT200)
\EN7      unbound          pt200   # <SM UNLABEL>-C   (PT200)
\EN8      unbound          pt200   # <SM UNLABEL>-CS  (PT200)
\EN9      unbound          pt200   # <LG UNLABEL>-U   (PT200)
\EN:      unbound          pt200   # <LG UNLABEL>-S   (PT200)
\EN;      unbound          pt200   # <LG UNLABEL>-C   (PT200)
\EN<      unbound          pt200   # <LG UNLABEL>-CS  (PT200)
\EN=      unbound          pt200   # <SCRL UP>-S      (PT200)
\EN>      unbound          pt200   # <SCRL UP>-C      (PT200)
\EN?      unbound          pt200   # <SCRL UP>-CS     (PT200)
\EN@      unbound          pt200   # <SCRL DOWN>-S    (PT200)
\ENA      unbound          pt200   # <SCRL DOWN>-C    (PT200)
\ENB      unbound          pt200   # <SCRL DOWN>-CS   (PT200)
\ENC      unbound          pt200   # <SCRL LEFT>-S    (PT200)
\END      unbound          pt200   # <SCRL LEFT>-C    (PT200)
\ENE      unbound          pt200   # <SCRL LEFT>-CS   (PT200)
\ENF      unbound          pt200   # <SCRL RIGHT>-S   (PT200)
\ENG      unbound          pt200   # <SCRL RIGHT>-C   (PT200)
\ENH      unbound          pt200   # <SCRL RIGHT>-CS  (PT200)
\ENI      unbound          pt200   # <UP>-S           (PT200)
\ENJ      unbound          pt200   # <UP>-C           (PT200)
\ENK      unbound          pt200   # <UP>-CS          (PT200)
\ENL      unbound          pt200   # <DOWN>-S         (PT200)
\ENM      unbound          pt200   # <DOWN>-C         (PT200)
\ENN      unbound          pt200   # <DOWN>-CS        (PT200)
\ENO      unbound          pt200   # <LEFT>-S         (PT200)
\ENP      unbound          pt200   # <LEFT>-C         (PT200)
\ENQ      unbound          pt200   # <LEFT>-CS        (PT200)
\ENR      unbound          pt200   # <RIGHT>-S        (PT200)
\ENS      unbound          pt200   # <RIGHT>-C        (PT200)
\ENT      unbound          pt200   # <RIGHT>-CS       (PT200)
\ENU      unbound          pt200   # <CANCEL>-U       (PT200)
\ENV      unbound          pt200   # <CANCEL>-S       (PT200)
\ENW      unbound          pt200   # <CANCEL>-C       (PT200)
\ENX      unbound          pt200   # <CANCEL>-CS      (PT200)
\ENY      unbound          pt200   # <DELETE>-C       (PT200)
\ENZ      unbound          pt200   # <DELETE>-CS      (PT200)
\EN[      unbound          pt200   # <PREV>-S         (PT200)  (aka <PG UP>)
\EN\\     unbound          pt200   # <PREV>-C         (PT200)  (aka <PG UP>)
\EN]      unbound          pt200   # <PREV>-CS        (PT200)  (aka <PG UP>)
```

```
\EN\^     unbound              pt200   # <NEXT>-S        (PT200)  (aka <PG DN>)
\EN_      unbound              pt200   # <NEXT>-C        (PT200)  (aka <PG DN>)
\EN`      unbound              pt200   # <NEXT>-CS       (PT200)  (aka <PG DN>)
\ENa      unbound              pt200   # <END/BEGIN>-C   (PT200)
\ENb      unbound              pt200   # <END/BEGIN>-CS  (PT200)
\EO|      unbound              pt200   # <TAB>-C/CS      (PT200)
\EO{      unbound              pt200   # <B-TAB>-C/CS    (PT200)
\EOy      unbound              pt200   # <STOP>-S        (PT200)
\EOz      unbound              pt200   # <STOP>-C        (PT200)
\EO!      pfkey1       pst100|pt200   # <F1>-U
\EO)      pfkey1       pst100|pt200   # <F1>-S
\EO1      pfkey1       pst100|pt200   # <F1>-C
\EO9      pfkey1       pst100|pt200   # <F1>-CS
\EO"      pfkey2       pst100|pt200   # <F2>-U
\EO*      pfkey2       pst100|pt200   # <F2>-S
\EO2      pfkey2       pst100|pt200   # <F2>-C
\EO:      pfkey2       pst100|pt200   # <F2>-CS
\EO#      pfkey3       pst100|pt200   # <F3>-U
\EO+      pfkey3       pst100|pt200   # <F3>-S
\EO3      pfkey3       pst100|pt200   # <F3>-C
\EO;      pfkey3       pst100|pt200   # <F3>-CS
\EO$      pfkey4       pst100|pt200   # <F4>-U
\EO,      pfkey4       pst100|pt200   # <F4>-S
\EO4      pfkey4       pst100|pt200   # <F4>-C
\EO<      pfkey4       pst100|pt200   # <F4>-CS
\EO%      pfkey5       pst100|pt200   # <F5>-U
\EO-      pfkey5       pst100|pt200   # <F5>-S
\EO5      pfkey5       pst100|pt200   # <F5>-C
\EO=      pfkey5       pst100|pt200   # <F5>-CS
\EO&      pfkey6       pst100|pt200   # <F6>-U
\EO.      pfkey6       pst100|pt200   # <F6>-S
\EO6      pfkey6       pst100|pt200   # <F6>-C
\EO>      pfkey6       pst100|pt200   # <F6>-CS
\EO'      pfkey7       pst100|pt200   # <F7>-U
\EO/      pfkey7       pst100|pt200   # <F7>-S
\EO7      pfkey7       pst100|pt200   # <F7>-C
\EO?      pfkey7       pst100|pt200   # <F7>-CS
\EO(      pfkey8       pst100|pt200   # <F8>-U
\EO0      pfkey8       pst100|pt200   # <F8>-S
\EO8      pfkey8       pst100|pt200   # <F8>-C
\EO@      pfkey8       pst100|pt200   # <F8>-CS
\EOA      pfkey11      pst100|pt200   # <PF1>-U
\EOO      pfkey11      pst100|pt200   # <PF1>-S
\EO]      pfkey11      pst100|pt200   # <PF1>-C
\EOk      pfkey11      pst100|pt200   # <PF1>-CS
\EOB      pfkey12      pst100|pt200   # <PF2>-U
\EOP      pfkey12      pst100|pt200   # <PF2>-S
\EO\^     pfkey12      pst100|pt200   # <PF2>-C
\EOl      pfkey12      pst100|pt200   # <PF2>-CS
\EOC      pfkey13      pst100|pt200   # <PF3>-U
\EOQ      pfkey13      pst100|pt200   # <PF3>-S
\EO_      pfkey13      pst100|pt200   # <PF3>-C
\EOm      pfkey13      pst100|pt200   # <PF3>-CS
\EOD      pfkey14      pst100|pt200   # <PF4>-U
\EOR      pfkey14      pst100|pt200   # <PF4>-S
\EO`      pfkey14      pst100|pt200   # <PF4>-C
```

```
  \EOn      pfkey14          pst100|pt200   # <PF4>-CS
  \EOE      pfkey15          pst100|pt200   # <PF5>-U
  \EOS      pfkey15          pst100|pt200   # <PF5>-S
  \EOa      pfkey15          pst100|pt200   # <PF5>-C
  \EOo      pfkey15          pst100|pt200   # <PF5>-CS
  \EOF      pfkey16          pst100|pt200   # <PF6>-U
  \EOT      pfkey16          pst100|pt200   # <PF6>-S
  \EOb      pfkey16          pst100|pt200   # <PF6>-C
  \EOp      pfkey16          pst100|pt200   # <PF6>-CS
  \EOG      pfkey17          pst100|pt200   # <PF7>-U
  \EOU      pfkey17          pst100|pt200   # <PF7>-S
  \EOc      pfkey17          pst100|pt200   # <PF7>-C
  \EOq      pfkey17          pst100|pt200   # <PF7>-CS
  \EOH      pfkey18          pst100|pt200   # <PF8>-U
  \EOV      pfkey18          pst100|pt200   # <PF8>-S
  \EOd      pfkey18          pst100|pt200   # <PF8>-C
  \EOr      pfkey18          pst100|pt200   # <PF8>-CS
  \EOI      pfkey19          pst100|pt200   # <PF9>-U
  \EOW      pfkey19          pst100|pt200   # <PF9>-S
  \EOe      pfkey19          pst100|pt200   # <PF9>-C
  \EOs      pfkey19          pst100|pt200   # <PF9>-CS
  \EOJ      pfkey20          pst100|pt200   # <PF10>-U
  \EOX      pfkey20          pst100|pt200   # <PF10>-S
  \EOf      pfkey20          pst100|pt200   # <PF10>-C
  \EOt      pfkey20          pst100|pt200   # <PF10>-CS
  \EOK      pfkey21          pst100|pt200   # <PF11>-U
  \EOY      pfkey21          pst100|pt200   # <PF11>-S
  \EOg      pfkey21          pst100|pt200   # <PF11>-C
  \EOu      pfkey21          pst100|pt200   # <PF11>-CS
  \EOL      pfkey22          pst100|pt200   # <PF12>-U
  \EOZ      pfkey22          pst100|pt200   # <PF12>-S
  \EOh      pfkey22          pst100|pt200   # <PF12>-C
  \EOv      pfkey22          pst100|pt200   # <PF12>-CS
  \EOM      pfkey9           pst100         # <PF13>-U      (PST100)
  \EOM      pfkey9                  pt200   # <F9>-U         (PT200)
  \EO[      pfkey9           pst100         # <PF13>-S      (PST100)
  \EO[      pfkey9                  pt200   # <F9>-S         (PT200)
  \EOi      pfkey9           pst100         # <PF13>-C      (PST100)
  \EOi      pfkey9                  pt200   # <F9>-C         (PT200)
  \EOw      pfkey9           pst100         # <PF13>-CS     (PST100)
  \EOw      pfkey9                  pt200   # <F9>-CS        (PT200)
  \EON      pfkey10          pst100         # <PF14>-U      (PST100)
  \EON      pfkey10                 pt200   # <F10>-U        (PT200)
  \EO\\     pfkey10          pst100         # <PF14>-S      (PST100)
  \EO\\     pfkey10                 pt200   # <F10>-S        (PT200)
  \EOj      pfkey10          pst100         # <PF14>-C      (PST100)
  \EOj      pfkey10                 pt200   # <F10>-C        (PT200)
  \EOx      pfkey10          pst100         # <PF14>-CS     (PST100)
  \EOx      pfkey10                 pt200   # <F10>-CS       (PT200)
# ^?        rubout_char      pst100         # <DEL>-ALL     (PST100)
# ^?        rubout_char             pt200   # <BACKSP>-C/CS  (PT200)
# ^Q        (flow-control)   pst100         # <PAUSE>-ALL   (PST100)
# ^Q        (flow-control)          pt200   # <STOP>-U       (PT200)
# ^S        (flow-control)   pst100         # <PAUSE>-ALL   (PST100)
# ^S        (flow-control)          pt200   # <STOP>-U       (PT200)
# (internal-to-term)         pst100         # <STOP>-S/C/CS (PST100)
```

```
# (internal-to-term)              pt200   # <STOP>-CS       (PT200)
# (internal-to-term)       pst100         # <DELETE>-C/CS  (PST100)
# (internal-to-term)       pst100         # <AUX SND>-CS   (PST100)
# (internal-to-term)              pt200   # <PRT SCN>-CS    (PT200)
# (internal-to-term)       pst100         # <CONFG>-U/S/CS (PST100)
# (internal-to-term)       pst100         # <MENU>-S/C/CS  (PST100)
# (internal-to-term)       pst100         # <RESET>-S/CS   (PST100)
# (internal-to-term)              pt200   # <CLEAR>-S/CS    (PT200)
# (internal-to-term)       pst100|pt200   # <ENTER>-C/CS
# (internal-to-term)              pt200   # <SETUP>-ALL     (PT200)
# (internal-to-term)              pt200   # <CHNG MODE>-ALL(PT200)

###############################################################################################
###
### PT45 bindings:
###
###############################################################################################
# ^C        submit          pt45|dm30|b150 # <ENTER>-ALL           (DIP switch selectable)
# ^D        submit          pt45|dm30|b150 # <ENTER>-ALL           (DIP switch selectable)
# ^H        rubout_char     pt45|dm30|b150 # <BACKSP>-ALL
# ^I        expand_wild     pt45|dm30|b150 # <TAB>-ALL
# ^J        submit          pt45|dm30|b150 # <ENTER>-ALL           (DIP switch selectable)
# ^J        submit          pt45|dm30|b150 # <RETURN>-ALL
# ^W        unbound         pt45|dm30|b150 # <SEND>-ALL
# \E        (Prefix)        pt45|dm30|b150 # <ESC>-U/S
  \E!       echo_raw        pt45|dm30|b150 # <PAGE>-ALL           Page Mode
  \E#       echo_raw        pt45|dm30|b150 # <PAGE>-ALL           Line Mode
# \E&       unbound         pt45|dm30|b150 # <FORMS>-C/CS         Forms Mode On
# \E(       echo_raw        pt45|dm30|b150 # <AUX ON>-U/S         AUX port on
# \E)       echo_raw        pt45|dm30|b150 # <AUX ON>-ALL         AUX/Parallel port off
  \E*       echo_raw        pt45|dm30|b150 #                     Monitor Mode Off
  \E,       unbound         pt45|dm30|b150 # <CE>-ALL            Clear Entry [FORMS]
# \E-       echo_raw        pt45|dm30|b150 # <AUX ON>-C/CS        Parallel port on
  \E.0      echo_raw        pt45|dm30|b150 # <SCRL UP>-U/S        Scroll Up
  \E.1      echo_raw        pt45|dm30|b150 # <SCRL UP>-C/CS       Next Page
  \E.2      echo_raw        pt45|dm30|b150 # <SCRL DOWN>-U/S      Scroll Down
  \E.3      echo_raw        pt45|dm30|b150 # <SCRL DOWN>-C/CS     Previous Page
# \E.4      echo_raw        pt45|dm30|b150 # <PRINT>-U/S          Parallel Print
# \E.5      echo_raw        pt45|dm30|b150 # <PRINT>-C/CS         Parallel Enable
# \E.5      echo_raw        pt45|dm30|b150 # <E-AUX>-C/CS         Parallel Enable
  \E.6      echo_raw        pt45|dm30|b150 # <HOME>-C/CS         Absolute Home
# \E0       echo_raw        pt45|dm30|b150 # <ASEND>-ALL
  \E:       echo_raw        pt45|dm30|b150 #                     Monitor Mode On
  \E;       unbound         pt45|dm30|b150 #                     CPU Message Deposit
# \E>       unbound         pt45|dm30|b150 # <B-TAB>-ALL         Back Tab
# \E@       unbound         pt45|dm30|b150 # <I CHAR>-ALL        Line Insert Mode Off
  \EA       prev_line       pt45|dm30|b150 # <UP>-ALL           Cursor Up
  \EB       next_line       pt45|dm30|b150 # <DOWN>-ALL         Cursor Down
  \EC       forward_char    pt45|dm30|b150 # <RIGHT>-ALL       Cursor Right
  \ED       back_char       pt45|dm30|b150 # <LEFT>-ALL        Cursor Left
  \EE       echo_raw        pt45|dm30|b150 # <CLEAR>-C/CS       Clear Screen
  \EH       echo_raw        pt45|dm30|b150 # <HOME>-U/S         Home
  \EJ       kill_line       pt45|dm30|b150 # <CLEAR EOP/EOF>-U/S  Clear to End of Page
  \EK       kill_line       pt45|dm30|b150 # <CLEAR EOP/EOF>-C/CS Clear to End of Line
  \EL       echo_raw        pt45|dm30|b150 # <I LINE>-ALL        Insert Line
  \EM       echo_raw        pt45|dm30|b150 # <D LINE>-ALL        Delete Line
```

```
  \EN       echo_raw       pt45|dm30|b150 # <LOCAL>-ALL          Off Line
  \EP       delete_char    pt45|dm30|b150 # <D CHAR>-U/S        Line Delete Character
# \EQ       unbound        pt45|dm30|b150 # <I CHAR>-U/S        Line Insert Mode On
# \EV       unbound        pt45|dm30|b150 #                      Reset
# \EW       unbound        pt45|dm30|b150 # <FORMS>-U/S         Forms Mode On
# \EX       unbound        pt45|dm30|b150 # <FORMS>-ALL         Forms Mode Off
  \EZ       unbound        pt45|dm30|b150 #                      Cursor Display On/Off
  \E`       delete_char    pt45|dm30|b150 # <D CHAR>-C/CS       Page Delete Character
# \Ea       unbound        pt45|dm30|b150 # <I CHAR>-C/CS       Page Insert Mode On
# \Ed       unbound        pt45|dm30|b150 # <A-SET>-U/S         Attribute Set
# \Ee       unbound        pt45|dm30|b150 # <A-SET>-C/CS       Attribute Clear
# \Eg       unbound        pt45|dm30|b150 # <M-LOCK>-ALL       Memory Lock On
# \Eh       unbound        pt45|dm30|b150 # <M-LOCK>-ALL       Memory Lock Off
# \Ej       echo_raw       pt45|dm30|b150 # <E-AUX>-U/S        AUX enable
# \Ek       echo_raw       pt45|dm30|b150 # <E-AUX>-ALL        AUX/Parallel disable
# \El       unbound        pt45|dm30|b150 #                      Start Blink
# \Em       unbound        pt45|dm30|b150 #                      Normal Video
# \En       echo_raw       pt45|dm30|b150 # <LOCAL>-ALL          On Line
  ^B^B      back_char      pt45|dm30|b150 #
  ^B\Ep     pfkey1         pt45|dm30|b150 # <F1>-U/S            f1
  ^B\Eq     pfkey2         pt45|dm30|b150 # <F2>-U/S            f2
  ^B\Er     pfkey3         pt45|dm30|b150 # <F3>-U/S            f3
  ^B\Es     pfkey4         pt45|dm30|b150 # <F4>-U/S            f4
  ^B\Et     pfkey5         pt45|dm30|b150 # <F5>-U/S            f5
  ^B\Eu     pfkey6         pt45|dm30|b150 # <F6>-U/S            f6
  ^B\Ev     pfkey7         pt45|dm30|b150 # <F7>-U/S            f7
  ^B\Ew     pfkey8         pt45|dm30|b150 # <F8>-U/S            f8
  ^B\Ex     pfkey9         pt45|dm30|b150 # <F9>-U/S            f9
  ^B\Ey     pfkey10        pt45|dm30|b150 # <F10>-U/S           f10
  ^B\Ez     pfkey11        pt45|dm30|b150 # <F11>-U/S           f11
  ^B\E{     pfkey12        pt45|dm30|b150 # <F12>-U/S           f12
  ^B\E|     pfkey13        pt45|dm30|b150 # <F13>-U/S           f13
  ^B\E}     pfkey14        pt45|dm30|b150 # <F14>-U/S           f14
  ^B\E~     pfkey15        pt45|dm30|b150 # <F15>-U/S           f15
  ^B\E^?    pfkey16        pt45|dm30|b150 # <F16>-U/S           f16
  ^B^Np     pfkey17        pt45|dm30|b150 # <F1>-C/CS           f1 control
  ^B^Nq     pfkey18        pt45|dm30|b150 # <F2>-C/CS           f2 control
  ^B^Nr     pfkey19        pt45|dm30|b150 # <F3>-C/CS           f3 control
  ^B^Ns     pfkey20        pt45|dm30|b150 # <F4>-C/CS           f4 control
  ^B^Nt     pfkey21        pt45|dm30|b150 # <F5>-C/CS           f5 control
  ^B^Nu     pfkey22        pt45|dm30|b150 # <F6>-C/CS           f6 control
  ^B^Nv     pfkey23        pt45|dm30|b150 # <F7>-C/CS           f7 control
  ^B^Nw     pfkey24        pt45|dm30|b150 # <F8>-C/CS           f8 control
  ^B^Nx     pfkey25        pt45|dm30|b150 # <F9>-C/CS           f9 control
  ^B^Ny     pfkey26        pt45|dm30|b150 # <F10>-C/CS          f10 control
  ^B^Nz     pfkey27        pt45|dm30|b150 # <F11>-C/CS          f11 control
  ^B^N{     pfkey28        pt45|dm30|b150 # <F12>-C/CS          f12 control
  ^B^N|     pfkey29        pt45|dm30|b150 # <F13>-C/CS          f13 control
  ^B^N}     pfkey30        pt45|dm30|b150 # <F14>-C/CS          f14 control
  ^B^N~     pfkey31        pt45|dm30|b150 # <F15>-C/CS          f15 control
  ^B^N^?    pfkey32        pt45|dm30|b150 # <F16>-C/CS          f16 control
# (ignored-by-Primos)      pt45|dm30|b150 # <LF>-ALL
# (internal-to-term)       pt45|dm30|b150 # <ESC>-C/CS
# (internal-to-term)       pt45|dm30|b150 # <CLEAR>-U/S        No operation
# (internal-to-term)       pt45|dm30|b150 # <RESET>-ALL        Internal Reset
```

Compiling, Saving, and Restoring the Bindings File

When you have finished setting up your bindings file, you must compile it to bind each sequence to the specified command. However, compiling the file makes those bindings effective for the current terminal session only. You have to recompile them the next time you want to use them. Instead, if you save the compiled version of the file, then you can restore the compiled version the next time you log in. This is faster than recompiling the bindings file again.

The compiled file is made up of two parts: the compiled bindings and the terminal capabilities. The bindings are the actual sequences bound to their corresponding commands. The capabilities are the characteristics of the particular type of terminal; baud rate, screen width, screen length, and so forth.

Compiling the Bindings File

To compile the bindings file, use the –BIND_TERM option in the following format:

ECL –TERMINAL_TYPE **terminal** –BIND_TERM **[bindings_file] [term_alias]**

–TERMINAL_TYPE specifies the type of terminal you are using (for example, PT45, PST100, PT200). If you specified bindings for different terminal types in your bindings file, ECL establishes only those bindings that apply to the specified terminal. If you do not specify **terminal**, ECL determines what your terminal type is by checking the global variable .TERMINAL_TYPE$. If your terminal type is not found in .TERMINAL_TYPE$, ECL establishes the capabilities of a generic 24-line by 80-column terminal.

–BIND_TERM binds each key sequence in **bindings_file** to the ECL command specified with it. The bindings contained in the file supplement the fundamental bindings that ECL automatically makes, or override existing fundamental bindings.

Reestablishing Fundamental Bindings

To reestablish the ECL fundamental bindings and to negate any specific bindings you might have previously created, issue the –BIND_TERM option without specifying a filename.

Saving a Compiled File

Issue the –SAVE_TERM option in the following format:

ECL –SAVE_TERM **compiled_file**

This command stores the compiled bindings in **compiled_file**. If you want to restore these bindings the next time you log in, **compiled_file** is the filename you must specify with the –RESTORE_TERM option.

```
OK, ECL -STERM SAVED_BINDINGS
Saved PT200 terminal capabilities and bindings (9600 baud). (EDIT_CMD_LINE)
OK,
```

Restoring a Compiled File

Use the –RESTORE_TERM option in the following format:

ECL –RESTORE_TERM compiled_file

This activates a compiled bindings file for the current session. If you always want the bindings in **compiled_file** to be in effect, consider including the ECL –RESTORE_TERM command in your login program.

```
OK, ECL -RTERM COMPILED_BINDINGS
Restored PT200 terminal capabilities and bindings (9600 baud). (EDIT_CMD_LINE)
OK,
```

Saving and Restoring Simultaneously

You can use both –STERM and –RTERM on the same command line. If you do so, the command line variable **compiled_file** is optional for –SAVE_TERM. –STERM and –RTERM together are used in conjunction with the –BPS, –TERMINAL_TYPE, and –BIND_TERM options, if you want to specify all these options simultaneously, as in the following example. (–BPS sets the terminal baud rate.)

```
OK, ECL -TTP PT200 -BTERM BINDINGS -RTERM BINDINGS -STERM -BPS 9600
Restored PT200 terminal capabilities and bindings (9600 baud). (EDIT_CMD_LINE)
OK,
```

ECL processes the above command line as follows:

1. If you specify –RTERM, ECL attempts to restore the compiled terminal characteristics and bindings in **compiled_file**. **compiled_file** contains the terminal type and baud rate with which it was compiled, and ECL checks that type and baud rate agree with those specified by –TTP and –BPS on the command line *before* completing the restore operation.
2. If the restore operation succeeds, ECL does not establish any more bindings by processing the –BIND_TERM option. In other words, the bindings in the existing compiled file are established, not those in the file specified by –BIND_TERM on the command line. Also, ECL ignores any –STERM option.
3. If the restore operation fails, either because **compiled_file** does not exist or **compiled_file**'s terminal type and baud rate do not match those specified on the command line, then terminal capabilities and bindings are compiled normally as if the restore operation were never requested.
4. If –STERM was also specified on the command line, the capabilities and bindings are saved in **compiled_file**. This allows you, on one command line, to restore compiled bindings for your terminal and (if they haven't yet been compiled) to automatically compile them, so that the restore operation will succeed the next time.

Note

If you change your terminal capability or bindings files, ECL notes that the compiled file is out-of-date and recompiles the out-of-date portion automatically. This requires that –BTERM be enabled (to verify the bindings portion of the compiled file) and that –TTP be enabled (to verify the capabilities portion of the compiled file) or –BPS (which implies –TTP).

What Are the Current Bindings?

You can use several methods to find out what a particular key sends, that is, what action a particular key is bound to.

The ECL command line option –WALLPAPER displays the current command bindings, including function key programs. If you want the display put into a file, use the following format:

ECL –WALL**PAPER [bindings_file]**

The display format that –WALLPAPER gives you is in the proper format, so that you can reload a bindings file produced in this manner using the –BIND_TERM option.

Also, you can use the command *explain_key* (Ctrl _). This command prompts for a command sequence and returns the command name bound to that sequence.

In addition, you can use the command *do_quote* (Esc '). This command self-inserts the characters that follow into the command line without further processing. Besides being used to quote many characters simultaneously, *do_quote* is also useful for displaying the character sequence that a key actually sends. For example, the *do_quote* command executed on the PT200 terminal displays **^[N!** when you press the PA1 key.

Function Key Codes

This section contains the codes that you must enter in a bindings file for function keys on PST100, PT200, and PT45 terminals. Table 6-2 lists function key sequences for the PST 100 and the PT200 terminals. Table 6-3 lists function key sequences for the PT45 terminal. If you are not using one of these terminals, consult the programmer's reference guide for your terminal to find out what codes to use, or use *do_quote* as explained above to construct such a table yourself.

TABLE 6-2
PST100 and PT200 Function Key Sequences

		Augmented With		
Key	*Unaugmented*	*Shift*	*Ctrl*	*Ctrl-Shift*
F1	\EO!	\EO)	\EO1	\EO9
F2	\EO"	\EO*	\EO2	\EO:
F3	\EO#	\EO+	\EO3	\EO;
F4	\EO$	\EO,	\EO4	\EO<
F5	\EO%	\EO-	\EO5	\EO=
F6	\EO&	\EO.	\EO6	\EO>
F7	\EO'	\EO/	\EO7	\EO?
F8	\EO(	\EO0	\EO8	\EO@
PF1	\EOA	\EOO	\EO]	\EOk
PF2	\EOB	\EOP	\EO\^	\EOl
PF3	\EOC	\EOQ	\EO_	\EOm

TABLE 6-2
PST100 and PT200 Function Key Sequences - Continued

		Augmented With		
Key	*Unaugmented*	*Shift*	*Ctrl*	*Ctrl-Shift*
PF4	\EOD	\EOR	\EO‘	\EOn
PF5	\EOE	\EOS	\EOa	\EOo
PF6	\EOF	\EOT	\EOb	\EOp
PF7	\EOG	\EOU	\EOc	\EOq
PF8	\EOH	\EOV	\EOd	\EOr
PF9	\EOI	\EOW	\EOe	\EOs
PF10	\EOJ	\EOX	\EOf	\EOt
PF11	\EOK	\EOY	\EOg	\EOu
PF12	\EOL	\EOZ	\EOh	\EOv
The next two keys appear on PST100s only.				
PF13	\EOM	\EO[	\EOi	\EOw
PF14	\EON	\EO\\	\EOj	\EOx
The next six keys appear on PT200s only.				
F9	\EOM	\EO[	\EOi	\EOw
F10	\EON	\EO\\	\EOj	\EOx
PA1	\EN!	\EN%	\EN)	\EN-
PA2	\EN"	\EN&	\EN*	\EN.
PA3	\EN#	\EN’	\EN+	\EN/
PA4	\EN$	\EN(	\EN,	\EN0

TABLE 6-3
PT45 Function Key Sequences

Key	*Unaugmented*	*Augmented With Ctrl*
F1	^B\Ep	^B^Np
F2	^B\Eq	^B^Nq
F3	^B\Er	^B^Nr
F4	^B\Es	^B^Ns
F5	^B\Et	^B^Nt
F6	^B\Eu	^B^Nu
F7	^B\Ev	^B^Nv
F8	^B\Ew	^B^Nw
F9	^B\Ex	^B^Nx
F10	^B\Ey	^B^Ny
F11	^B\Ez	^B^Nz
F12	^B\E{	^B^N{
F13	^B\E|	^B^N|
F14	^B\E}	^B^N}
F15	^B\E~	^B^N~
F16	^B\E^?	^B^N^?

Display Manager

The ECL display manager, as the term suggests, controls the manner in which terminal output is displayed on the screen. The display manager efficiently maintains an accurate display of the command line, even if it wraps to more than one line on the screen. The display manager also optimizes cursor movement and does not update the screen until typeahead has been processed. Thus, inserting characters at the beginning of a long command line is relatively quick, even across a network connection.

The display manager is controlled by the ECL$LIB subroutine, which replaces the PRIMOS CL$GET routine when ECL is initialized. Also, in order that ECL be able to use different terminals, the display manager also references a database called TERMINFO (discussed below) to get information about the capabilities of a particular terminal.

Terminal Requirements for the Display Manager

Here are the minimum requirements for your terminal in order for the display manager to properly operate:

- A character placed in the last column of the screen must wrap the cursor to the first column of the next line. (If not, then ECL –WIDTH 0 must be specified.)
- A linefeed (Ctrl J) character, or a screen wrap off the bottom line of the display, must cause a new blank line to scroll on the screen.
- A backspace (Ctrl H) character must move the cursor one position to the left without erasing anything.
- A backspace (Ctrl H) must wrap from the first column on a line to the last column of the previous line.

Other Requirements: ECL assumes, for a generic terminal, a default terminal display of 24 lines by 80 columns (although you can change this with ECL's –HEIGHT and –WIDTH command line options, or with a TERMINFO entry). The display manager normally requires that ECL perform its own character echoing (as EMACS does), rather than PRIMOS.

Specifying Terminal Type

As mentioned above, the display manager references a database called TERMINFO to get information about the capabilities of a particular terminal. But first, you must tell ECL what type of terminal you are using. You can do this in two ways:

1. Using the –TERMINAL_TYPE **[type]** command-line option. This overrides any types defined in the global variable .TERMINAL_TYPE$.
2. Using the global variable .TERMINAL_TYPE$ (the same variable used by EMACS). ECL checks this variable whenever –TERMINAL_TYPE is invoked with no argument.

If ECL cannot find an appropriate entry in TERMINFO for the specified terminal, or there is some other error, ECL defaults to its generic terminal capabilities. Once terminal capabilities and bindings are established for a particular terminal, you can then save and restore them in compiled form using ECL's –SAVE_TERM and –RESTORE_TERM options. (See the section ECL Bindings earlier in this chapter.)

Prompt Handling

In order for ECL's display manager to properly optimize its display operations, it must know about the prompt. Furthermore, prompts must consist only of printable characters (that is, they cannot contain terminal-dependent sequences, for example, that enable blink mode).

By default, ECL recognizes the standard PRIMOS RDY prompts, displaying their brief or long forms as appropriate. ECL also has its own internal set of brief-form prompts independent of those specified by the RDY command; these default to `OK,` `ER!,` and `OK,` for the ready, error, and warning brief prompts, respectively. You specify these prompts with the ECL command-line options –RB, –EB, and –WB. Using these options causes ECL to use its own set in place of the PRIMOS set; long-form prompts are unaffected.

The advantage of using ECL's own set of prompts is that the number-sign (#) character, with an ECL prompt, acts as a placeholder. # instructs ECL to substitute the current command history entry number in place of the # character when displaying the prompt. (To display a number-sign in your prompt, use two number-signs in tandem: ##.) This is the recommended way of determining the command history entry number for the *goto_line* command, discussed in the section ECL Commands Reference. To enable prompt numbering, put the number sign and any other characters you wish to use inside single quotation marks when customizing your prompts:

```
OK, ECL -ON -READY_BRIEF '# >> ' -ERROR_BRIEF '# !? '
001 >> _
```

Now both you and ECL can keep track of your commands in the command history by their number in the history. For more information about the command history, see the *PRIMOS User's Guide.*

The TERMINFO Database

The TERMINFO database is a text file that contains descriptions called **entries.** Each of these entries lists a set of capabilities for a terminal. Entries are compiled to produce individual files each describing a single terminal. The format and capabilities of Prime TERMINFO are documented fully in the AT&T™ System V (and PRIMIX Version 3.0) edition of the *Unix™ Programmer's Manual.*

TERMINFO itself resides in the top-level directory TERM*. Your System Administrator determines if you are using the TERMINFO database or an alternative database.

Note

This TERMINFO database is in the public domain and is made available to the user by courtesy of Prime Computer, Inc. Prime makes no representations or warranties whatsoever regarding this database, or the ability of any Prime software, when combined with this database, to operate on any terminals other than Prime terminals. Prime also disclaims any obligations to maintain or support this database or any similar database now or in the future.

Table 6-4 shows how the ECL display manager uses the following terminal capabilities.

TABLE 6-4
ECL TERMINFO Capabilities

Name	*Type*	*Full Name*	*Function*
am	bool	*auto_right_margin*	Wraps at right margin
bel	str	*bell*	Rings bell
bw	bool	*auto_left_margin*	Wraps back at left margin
cols	$	*columns*	Number of columns in a line
cr	str	*carriage_return*	Carriage return
cub1	str	*cursor_left*	Moves cursor left one space
cubw†	str	*cursor_left_wrap*	Wraps cursor to the left
cud1	str	*cursor_down*	Moves cursor down one line
cuf1	str	*cursor_right*	Moves cursor right one space
cuu1	str	*cursor_up*	Moves cursor up one line
dcd1†	str	*delete_char_display*	Deletes one character to end of display
dch1	str	*delete_char*	Deletes one character
ed	str	*clr_eos*	Clears to end of display
el	str	*clr_eol*	Clears to end of line
ht	str	*tab*	Tabs to next column
iddd†	str	*insert_delete_display*	Insert and delete in display mode
idde†	str	*insert_delete_line*	Insert and delete in line mode
in	bool	*insert_null_glitch*	Distinguishes nulls in insert mode
ind	str	*scroll_forward*	Scrolls forward
ip	str	*insert_padding*	Insert delay after an inserted character
ipd†	str	*insert_eos_padding*	Insert delay for insert to end of display
it	#	*init_tabs*	Tabs exist every # spaces
lines	#	*lines*	Number of lines on a screen or page
lis1†	str	*line_oriented_init1*	Initialization string for ECL
mir	bool	*move_insert_mode*	Allows movement in insert mode
pad	str	*pad_char*	Pad delay character (rather than null)
rlom†	str	*line_oriented_end*	Exits line-oriented (ECL) mode
rmir	str	*exit_insert_mode*	Exits insert mode
slom†	str	*line_oriented_start*	Enters line-oriented (ECL) mode
smid†	str	*enter_insert_display*	Enters insert-in-display mode
smir	str	*enter_insert_mode*	Enters insert or insert-in-line mode
xenl	bool	*eat_newline_glitch*	Ignores newline after 80 columns
xt	bool	*teleray_glitch*	Has Teleray tabs

†Non-standard TERMINFO extensions for Prime ECL.

Using TERMINFO

The display manager makes certain assumptions about several terminal capabilities. It assumes that

- **am** is specified (true)
- **bel** is an ASCII **BEL** (Ctrl G)
- **cr** is an ASCII **CR** (Ctrl M)
- **cub1** is an ASCII **BS** (Ctrl H)
- **cud1** is an ASCII **LF** (Ctrl J)
- **ht** is an ASCII **HT** (Ctrl I)
- **in** is not specified (false)
- **ind** is an ASCII **LF** (Ctrl J)
- **xt** is not specified (false)

If these assumptions are contradicted by the capabilities specified, ECL produces the following warning message:

```
Note: xxx terminal may not be properly supported. (EDIT_CMD_LINE)
```

The **lines** and **cols** capabilities may be overridden by ECL's –HEIGHT and –WIDTH options, respectively.

Backspace and Wrapping: If the terminal has **bw**, then **cub1** is used as a backspace that wraps backward from the first column to the last column of the previous line; otherwise, specify **cubw**. The display manager uses **cuu1** to move up one line if neither **bw** nor **cubw** have been specified.

TERMINFO guidelines state that **cuf1** may not necessarily wrap from the end of a line to the beginning of the next with **am** specified; however, ECL assumes this is possible.

Using the Clearing Capabilities: If your terminal supports both **ed** and **el**, it is best to specify both, because **ed** is usually slower and the display handler uses the faster of the two. The TERMINFO guidelines state that **ed** is only defined from the first column of a line, but ECL assumes it is valid from any column.

Optimization Using Insert Modes: For the display manager to make optimizations by using the terminal's insert character modes, these modes must operate within the entire display, not just the current line, and **mir** must be specified (true).

Some terminals have separate sequences for entering insert-in-line mode (**smir**) and insert-in-display mode (**smid**). Other terminals have one sequence to enter either insert mode (given as **smir**), coupled with a separate mode to indicate the extent of both insertion and deletion (line or display). Enabling the insert/delete-in-display extent is given as **idde**; reverting to the insert/delete-in-line extent as **iddd**. If specified, the **idde** capability is sent to the terminal every time ECL$LIB is called upon to read a new command line. Regardless of which insert mode extent is entered, exiting either insert mode is accomplished typically by using the single sequence **rmir**.

Optimization Using Delete Modes: As with the insert modes described above, the display manager makes optimizations by using the terminal's delete character modes to operate within the entire display, not just the current line, and **mir** must be specified (true).

For character deletion, some terminals have separate sequences for deleting a character in-line (**dch1**) and in-display (**dcd1**). Others use **idde** and **iddd** to specify the deletion extent of **dch1**. Padding delays for character insertion and deletion are specified as **ip**, with **ipd** specified if the delay is different for a display extent insertion or deletion.

Tab Stops: The **lis1** capability is an initialization string sent to the terminal once by ECL, *not* by ECL$LIB. You can use it to establish tab stops every **it** character positions so the **it** capability can be defined and used. If a terminal does support tabs somehow, then define the **it** capability; this significantly improves the display manager's performance. Most terminals power up with tab stops every 8 columns (**it#8**).

Changing the Video Display: You can use the **slom** and **rlom** capabilities to establish some terminal-specific visual mode for use within ECL$LIB (while entering a command line). Use this, for example, to change cursor shape or enable/disable reverse video to highlight the command line. The display manager initially sends **slom** immediately after **idde**, both of which occur before any prompt is displayed by ECL$LIB. The **rlom** capability is sent whenever ECL$LIB is exited (as with command submission) or when the display manager is suspended (as when displaying the version from the version command). In the latter case, the display manager restores **slom** when resumed, just before redisplaying the command line.

Disabling a Linefeed: The **xenl** capability indicates that the terminal ignores a newline (linefeed) after wrapping to a new line. Certain terminals exhibit this behavior.

ECL Commands Reference

The commands documented in this section, also called fundamental bindings, are ordered alphabetically by command name. Each command name is accompanied on the same line by the appropriate control or escape sequence. In the discussion of each command, any reference to numeric argument **n** refers to the *esc_digit* command used immediately before the listed command.

abort_cmd Ctrl G

Aborts your last input command. *abort_cmd* sets the return error code to -1 and returns the null string to the caller; this is useful when, for example, you wish to abort from a forward search. When used at command level (at the ECL prompt), *abort_cmd* always changes your position in the stack history to the end; that is, one position after the last command you issued. The numeric argument is not applicable and is ignored.

back_char Ctrl B
Moves the cursor back **n** characters in the command line.

back_word Esc B
Moves the cursor back **n** words, leaving the cursor positioned on the first character of that word. A **word** consists only of alphanumeric and underscore characters. The –ENTRY option to ECL modifies this definition to include all characters possible in a PRIMOS entryname, including wildcards. For more information about the –ENTRY option, see the discussion on the ECL command in Chapter 2.

begin_line Ctrl A
Moves the cursor to the beginning of the command line (the first character position after the prompt). The numeric argument is ignored.

collect_macro Esc (
Starts collecting keystrokes into a macro program that, once completed with the *finish_macro* (or *submit*) command, can be reexecuted with the *execute_macro* command. Keyboard macros have the ability to collect responses to prompts like those generated by the search commands. Only one keyboard macro, limited to 200 keystrokes, can be programmed at a time. The bell rings if a keyboard macro is already being collected when this command is issued. The numeric argument is ignored.

copy_region Esc W
Copies the contents of the region delimited by the cursor and mark positions (see the *mark* command) into the kill buffer. When the value of **n** is 4, *copy_region* copies the entire command line into the kill buffer. This command is similar to the *kill_region* command, except that the *kill_region* command also deletes the text from the command line.

delete_char Ctrl D
Deletes one character (or **n** characters if *esc_digit* is used) to the right of the cursor on the command line, starting with the character under the cursor. Deleted characters are not saved in a kill buffer.

delete_word Esc D
Kills forward one word (or **n** words if *esc_digit* is used) starting from the current position. The deleted text is placed in a kill buffer. See *back_word* for the definition of a word.

do_echo Esc @
Echoes **n** characters (including Ctrl P) directly to the terminal without including them in the command line. Issue the *do_echo* command immediately before typing the characters that you do not want ECL to process. Echoing is canceled when you repeat the original character sequence bound to *do_echo*. This is a useful way to cause local terminal operations, such as scrolling, to be performed without having the characters interpreted as being part of some command.

do_quote Esc '
Inserts **n** characters that follow it (including Ctrl P) into the command line without further processing. Quoting is canceled when you repeat the original character sequence bound to *do_quote*. *do_quote* is useful to quote many characters all at once rather than having to use the *quote* command for every character individually. See the discussion of the *quote* command later in this section.

downcase_word Esc L
Moves the cursor forward **n** words while simultaneously converting all uppercase letters to their lowercase equivalents. See the *back_word* command earlier in this section for the definition of a word.

echo_raw *[not bound]*

Echoes the bound character sequence directly to the terminal without including it in the command line. Use *echo_raw* to bind labeled terminal keys that perform local screen operations such as scrolling back pages or local cursor movement.

end_line Ctrl E

Moves the cursor to the end of the command line. The numerical argument is ignored.

esc_digit Esc **n**

Specifies how many times the next ECL command is to be executed. The numeric argument **n** defaults to one (1) for each command. *esc_digit* is a two-part sequence. First, press the Esc key. Second, press the decimal digit or digits desired. If you enter more than one decimal digit, they continue the count specification; for example, Esc 2 4 specifies a count of 24. (In this case, the next ECL command you entered would be executed 24 times.) This command is similar to the *multiplier* command, discussed later in this section.

exchange_mark Ctrl X Ctrl X

Exchanges the cursor position with the mark position (see the *mark* command). The actual text region delimited by these two positions remains unchanged.

execute_macro Esc E

Executes the most recent keyboard macro you defined by using the *collect_macro* and *finish_macro* commands. You can execute the previous keyboard macro while collecting a new one in order to include it within the new macro program. The bell rings if the keyboard macro program is null. To view the contents of the keyboard macro program, use the *explain_key* command for a sequence bound to *execute_macro*. The numeric argument, **n**, is applied only to the first command within the keyboard macro program.

expand_abbrev Esc A

Replaces a token with its abbreviation expansion. (The cursor must be within the token.) Any PRIMOS global variable references are also expanded; to prevent this, use a numeric argument (**n**) of 0. If **n** is 4, the entire command line (not beginning with ~) is expanded for abbreviations. If **n** is 16, the entire command line is expanded for global variables and abbreviations. In all of the above cases, the mark position is set at the beginning of the expansion and the cursor is placed at the end. (See the *mark* command later in this section.) Be aware that the token or command line replaced by the expansion is not implicitly saved in a kill buffer. Tokens are delimited by any characters in the set `>, ; [ ] ( )`.

expand_wild Ctrl I

Performs an automatic pathname completion on a partially completed pathname. (The cursor must be within the partially completed pathname.) The completion occurs as if you had used wildcards at the end of the partial pathname; however, ECL's –WILD_TAIL option makes wildcarding occur at the current cursor position within the pathname. If **n** is 1, the partially completed pathname is expanded as far as possible without the pathname becoming ambiguous. If this is already the case, or if **n** is 4, then the menu list of possible completions is displayed without affecting the partially completed pathname in the command line. You can make ECL process abbreviation expansion before pathname completion by using the option –WILD_ABBREV. The mark position is always set to the beginning of the completed pathname (see the *mark* command). See the section Automatic Pathname Completion in this chapter for more information on pathname completion.

expand_wild_menu Esc Ctrl 1

Selects the **n**th entry corresponding to the number in the displayed menu list of the possible pathname completions generated by the *expand_wild* command. *expand_wild_menu* substitutes the entry in the numbered list for the partial pathname in the command line. This command is typically used immediately after the *expand_wild* command.

explain_key Ctrl _

Prompts for a key sequence and displays the command name to which that sequence is bound. This is useful for determining what terminal function keys are bound to, and for debugging binding-related problems. This command is never collected within a keyboard macro (see *collect_macro*). The numeric argument, while ignored, is retained for the next command entered.

extend_command Esc X

Executes an ECL command at the **Command:** prompt. The command can be any command listed in this section. The use of the numeric argument depends on the specified command. Executing *extend_command* twice reexecutes the most recent command rather than reprompting.

finish_macro Esc)

Completes a keyboard macro being collected with the *collect_macro* command. The macro replaces the previous keyboard macro, if it exists, and can be executed with the *execute_macro* command. The bell is rung if no keyboard macro was being collected when this command is issued. The numeric argument is ignored.

forward_char Ctrl F

Moves the cursor forward **n** characters toward the end of the command line.

forward_search Ctrl S **or** Esc S

Prompts for a string, and searches **n** times in the forward direction (toward the end of the command history) for that string in each command of the command history. If the string is found, the corresponding command becomes the current command; if not, the bell rings. If the current command line is null, the search starts with the hidden command in the history; otherwise, the search ends with this hidden command.

The search string may begin, end, or both begin and end with a quoted newline character ('Ctrl J'), which instructs ECL to search for lines beginning (and/or ending) with the search string. A newline character is quoted using the *quote* command followed by pressing Return on your terminal.

The search facility maintains its own ring of search strings. This is analogous to the command history, but the search ring is accessible only from within a search prompt. Entering a null string in response to the search prompt instructs ECL to use the most recent search string. The most recent string is also used automatically when a search command is reexecuted with the *reexecute* command. By default, searching is not case-sensitive, but this can be changed by using the –CASE_SEARCH option.

forward_word Esc F

Moves the cursor forward one word (or **n** words if *esc_digit* is used), leaving the cursor positioned after the last character of that word. See *back_word* for the definition of a word.

goto_line Esc G

Moves the cursor directly to the entry in the command history specified by the numeric argument **n**. For example, Esc 5 Esc G moves the cursor to command history entry number 5. That entry then becomes the current command line being edited. If **n** is associated with the current command, the hidden command from the command history is recalled. An **n** value of 0 repositions to an empty command at the end of the command

history (the top of the history stack). Use the *refresh* command to determine the entry number associated with commands in the history list. You can also specify (with arguments to the ECL –RB/–EB/–WB options) a command-level prompt displaying the current history entry number for each command. See the discussion about ECL in Chapter 2 for more details about setting up such a prompt.

kill_line Ctrl K

Kills the remainder of the line from the cursor to the end of the command line. A numeric argument of 0 or greater than 1 instructs ECL to instead kill from the beginning of the line to the cursor. The deleted text is placed into a kill buffer for possible recall with the *yank* or *yank_replace* command.

kill_region Ctrl W

Kills the contents of the region delimited by the cursor and mark positions (see the *mark* command). A numeric argument of 4 instructs *kill_region* to kill the entire command line instead. The deleted text is placed into a kill buffer.

mark Ctrl @

Sets the mark position at the current cursor position. From then on, the mark acts as a hidden cursor bound to the character to its left. Deletions and insertions that move this character have a corresponding effect on the mark. The numeric argument, if given, is ignored.

The mark and cursor positions delimit a region of text operated upon by the *copy_region* and *kill_region* commands. It doesn't matter which end is mark and which is cursor, the region is the same. The hidden mark position is revealed by using the *exchange_mark* command. Only the most recent setting of the mark position is kept.

multiplier Ctrl U

Multiplies the numeric argument, **n**, for the next command by 4. The numeric argument defaults to one (1) for each command. *multiplier* multiplies **n** by four each time it is invoked. For example, executing the command sequence "*multiplier multiplier forward_char*" moves the cursor forward 16 (4 x 4) characters.

However, if the *multiplier* command is followed by a sequence of decimal digits, that sequence specifies the exact number of times to execute the command. For example, Ctrl U 1 2 specifies a count of 12. See also the *esc_digit* command described earlier in this section as an alternative for specifying a prefix count.

next_line Ctrl N

Moves the cursor toward the end of the command history by **n** entries. The command line reached there becomes the current command line being edited. A numeric argument of zero recalls the command at the current place within the command history hidden by the current command. Relative movement within the command history is affected by the ECL options –STACK / –NO_STACK and –STICK / –NO_STICK. See Chapter 2 for more information about these options.

password Esc P

Instructs ECL to neither echo nor save the command line input immediately following due to its sensitive content. The input is saved neither in the command history nor in a clean COMO file. The *password* command must be issued before typing any of the command line; the bell rings if the command line already contains any text. When the prompt includes the current history event number, this is replaced with three number-sign characters (###) to indicate that the command is not being saved in the history. The password is actually retrieved via the PRIMOS routine CL$GET, so only the PRIMOS erase and kill characters are observed during its entry. The numeric argument is ignored.

pfkey1 - pfkey512 *[not bound]*

Executes the program bound to the corresponding Programmable Function command (PFkey) sequence. The numeric argument, **n**, is effective only on the first command within the programmed sequence. If no sequence has been programmed, the terminal bell rings, indicating the PFkey is unprogrammed. Programming PFkeys is discussed in the section ECL Key Bindings.

prev_line Ctrl Z

Moves the cursor one entry (or **n** entries if *esc_digit* is used) toward the beginning of the command history. The command line saved there becomes the current command line being edited. Executing *prev_line* with no numeric argument retrieves your last command. An argument of zero recalls the command at the current place within the command history hidden by the current command. Relative movement within the command history is affected by the ECL options –STACK / –NO_STACK and –STICK / –NO_STICK, which are discussed in Chapter 2.

quote Ctrl Q **or** Esc Q

Quotes the next character typed in (including Ctrl P) for literal insertion one time (or **n** times if *esc_digit* is used). This permits the insertion of characters into the command line that are normally bound to nonprintable commands. If there are many characters to quote, use the *do_quote* command, discussed earlier in this section.

reexecute Ctrl C

Reexecutes the previous command one time (or **n** times if *esc_digit* is used). If the previous command has a numeric argument associated with it, the two counts are multiplied together. *reexecute* itself is never considered the previous command; that is, reexecuting *reexecute* acts upon the command prior to the first reexecute.

refresh Ctrl L

Reinitializes the terminal and redisplays the current command line. Use *refresh* to clean up the terminal display or to reset your terminal. This command has a default numeric value of 1; if **n** is greater than 1, the previous **n-1** commands in the history are displayed before the current command line is redisplayed. A numeric value of zero reveals the current history event number and any history command hidden by the current command line.

reverse_search Ctrl R **or** Esc R

Prompts for a string, and searches one time (or **n** times if *esc_digit* is used) in the reverse direction (toward the beginning) for that string in the command history. If found, the corresponding command becomes the current command; otherwise, the bell rings. *reverse_search* always starts with the previous command in the history and includes the hidden command last. See the *forward_search* command for more information on searching.

rubout_char Ctrl H **or** DEL

Deletes one character (or **n** characters if *esc_digit* is used) to the left on the command line, starting with the character preceding the cursor. Deleted characters are not saved in a kill buffer.

rubout_word Esc H **or** Esc DEL

Kills backward one word (or **n** words if *esc_digit* is used), starting from the current position. Deleted text is placed in a kill buffer. See *back_word* for the definition of a word.

self_insert *[printable characters]*

Inserts the last character of the bound sequence into the command line at the current position one time (or **n** times if *esc_digit* is used).

submit Ctrl J

Submits the entire command line to the process that called ECL$LIB. (Return is usually bound to the *submit* command.) The cursor can be anywhere on the command line. The *submit* command also implicitly finishes a keyboard macro if the macro was started at this level. If **n** is greater than one, *submit* toggles the command history behavior between ring and stack for the submitted command; for example, this causes ECL to use –NO_STACK if –STACK is in effect, and vice versa. An **n** value of 16 prevents the submitted command from being saved in the command history.

toggle_case Ctrl ^

Moves the cursor forward **n** characters while simultaneously toggling the case of each character. If the character is an uppercase letter, it is converted to its lowercase equivalent, and vice versa.

twiddle Ctrl T

Reverses the order of the two characters preceding the cursor on the command line. The numeric argument is ignored.

unbound *[everything not otherwise bound]*

Disables any binding for this sequence; the sequence is ignored and the terminal bell rings. Everything not bound to another command is implicitly bound to *unbound*. Explicitly binding to *unbound* can be useful in order to disable certain labeled terminal keys that can cause unpredictable results if echoed. The numeric argument is ignored.

upcase_word Esc U

Moves the cursor forward **n** words while converting all lowercase letters to their uppercase equivalents. This is the converse of the *downcase_word* command. See *back_word* earlier in this section for the definition of a word.

version Ctrl V

Displays the current version number for ECL$LIB.

yank Ctrl Y

Yanks back the text saved in the previous kill buffer **n**, inserting it into the command line at the current position. If **n** is 1, the current kill buffer, containing the text most recently deleted, is yanked. The mark position is always set to the beginning of the yanked text (see the *mark* command earlier in this section).

yank_replace Esc Y

Functions like the *yank* command except that most recently yanked text is replaced by the text of the indicated kill buffer. This permits you to search the kill ring for the deleted text of interest. The mark position is always set to the beginning of the yanked text (see the *mark* command earlier in this section).

A RVEC Parameters

Meaning of RVEC Parameters

The commands RESTOR, RESUME, SAVE, PM, and START process a group of optional parameters associated with the PRIMOS RVEC vector. These parameters are stored on disk for every runfile (executable program).

Initial values for the RVEC parameters are usually specified in the PRIMOS SAVE command, or by the LOAD or SEG SAVE command, when the program was stored on disk.

Each parameter is a 16-bit halfword, represented by as many as six octal digits.

Parameter	*Memory Location*	*Definition*
SA	—	Starting Address (first memory word used by program)
EA	—	Ending Address (last memory halfword used by program)
PC	7	P Register (Program Counter)
A	1	A Register (Arithmetic)
B	2	B Register (Arithmetic)
X	0	Index Register
Keys	—	Status keys associated with INK and OTK instructions

The RVEC parameters are optional in the command string. Any item that is specified replaces the previous value in RVEC, which is saved with the program. Thus, for any parameters that are not specified, the value previously stored in RVEC is saved with the program.

Slash Convention

An ordinal value followed by a slash (/) and a value can be used to set a selected octal parameter without setting other octal parameters. For example, given the command format

RESUME pathname [pc] [a] [b] [x] [keys]

the command

```
R FILNAM 2/1000
```

skips the first two RVEC parameters and sets the value of the third (b) to 1000_8.

Supplying RVEC Parameters

RVEC parameters specified in RESUME or START commands replace the previous values in RVEC. Also, when a program returns to PRIMOS through the EXIT subroutine, RVEC is loaded from the processor values in effect at the time of exit. Only the SAVE command alters the values of RVEC stored on disk with the program.

The RESTOR command returns a program from disk to memory and loads the SAVE parameters into RVEC in preparation for a START command.

The RESUME command combines the functions of RESTOR and START.

The PM command lists the current values of the RVEC parameters.

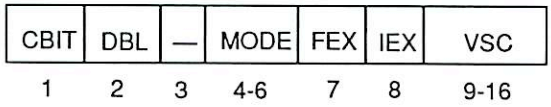
External commands have RVEC parameters that can be modified at the time the command is started (for example, PMA filename 1/740). Providing RVEC parameters to a command that does not need them causes unpredictable results.

Meaning of the Keys Parameter

The RVEC parameter **keys** refers to the processor status keys handled by the INK and OTK instructions. (For these instructions, see the *System Architecture Reference Guide*.) **keys** is represented by a single 16-bit word in either the Keys (SR) or Keys (VI) format. S-mode and R-mode programs use the Keys (SR) format; V-mode and I-mode programs use the Keys (VI) format.

Keys (SR) Format

The keys format for S and R modes is as follows:

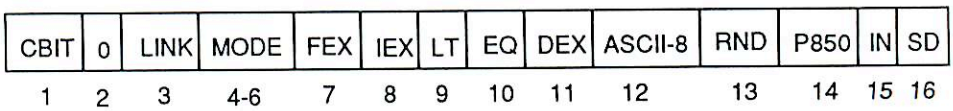


The mnemonics and meanings of the bits are as follows:

Bit	*Description*
CBIT (Bit 1)	Set by error conditions in arithmetic operations and by shifts
DBL (Bit 2)	Specifies the arithmetic mode: 0 Single precision 1 Double precision
– (Bit 3)	Reserved for future use
MODE (Bits 4-6)	Specifies the current addressing mode: 000 16S 001 32S 010 64R 011 32R 100 32I 101 Unused 110 64V 111 Unused
FEX (Bit 7)	Floating-point exception: 0 Set C bit to 1 and invoke fault handler on error 1 Set C bit to 1 only on error
IEX (Bit 8)	Integer exception: 0 Set C bit to 1 only on error 1 Set C bit to 1 and invoke fault handler on error
VSC (Bits 9-16)	Visible Shift Count: bottom half of the floating-point exponent

Keys (VI) Format

Process status information is available in a 16-bit register known as the Keys. The register may be referenced by the LPSW, TKA, and TAK instructions.



The mnemonics and meanings of the bits are as follows:

Bit	*Description*
CBIT (Bit 1)	Set by error conditions in arithmetic operations and by shifts
– (Bit 2)	Must be zero
L (Bit 3)	Set by an arithmetic or shift operation except IRS, IRX, DRX. Equal to carry out of the most significant bit (Bit 1) of an arithmetic operation. Valuable for simulating many precision operations and for performing unsigned comparisons following a CAS or a SUB
MODE (Bits 4-6)	Specifies the current addressing mode: 000 16S 001 32S 010 64R 011 32R 100 32I 101 Unused 110 64V 111 Unused
FEX (Bit 7)	Floating-point exception enable/disable: 0 Set C bit to 1 and invoke fault handler on error 1 Set C bit to 1 only on error
IEX (Bit 8)	Integer exception enable/disable: 0 Set C bit to 1 only on error 1 Set C bit to 1 and invoke fault handler on error
LT (Bit 9)	Less Than condition code: reflects the extended sign of the result (before truncation, if overflow); set to 1 if result is negative
EQ (Bit 10)	Equal To condition code: reflects the state of the result (after truncation, if overflow); set to 1 only if result is zero

DEX (Bit 11) Decimal exception enable/disable:

0	Set C bit to 1 only on error
1	Set C bit to 1 and invoke fault handler on error

ASCII-8 (Bit 12) ASCII character representation: specifies whether set or reset ASCII characters are generated (disregarded on the 2250 and earlier machines).

0	Most significant bit of characters is 1 (set format)
1	Most significant bit of characters is 0 (reset format)

RND (Bit 13) Floating-point round: specifies the form of rounding in floating-point operations (disregarded on the 2250 and earlier machines).

0	No rounding
1	Rounding

P850 (Bit 14) P850 bit used only by the P850 processor: set/cleared only by process exchange

IN (Bit 15) In dispatcher: specifies if the current process associated with the register is in the dispatcher

0	Process is in the dispatcher
1	Process is not in the dispatcher

SD (Bit 16) Save done: specifies if PXM (process exchange mechanism) has saved the values of the current register set

0	Save must be done before this register set can be used
1	Save has been done and this register set is available

B DUMP_STACK Format

DUMP_STACK Command

The DUMP_STACK command, explained in Chapter 2 of this guide, traces the sequence of calls and returns by which the user's process arrived at its current state. The user's command stack preserves machine states for internal commands, condition frames, and fault frames. In addition, the most recent activation of a static-mode program or command is preserved on the static-mode stack. DUMP_STACK displays the stack dump on the terminal or writes it to a command output file. Because DUMP_STACK is an internal command, it does not overwrite the static-mode stack, thereby permitting reentry into the faulting program.

Format of Stack Frames

DUMP_STACK lists each stack frame in the following general format:

(nn) offset: Owner= proc-name (LB= owner-lb).
Called from pcl-addr; returns to return-addr.

The meaning of each parameter is listed below. See the *System Architecture Reference Guide* for an explanation of the registers and rings involved.

Parameter	*Definition*
nn	Frame index number of the stack frame
offset	The word number in the current stack segment where the stack frame of this activation begins
proc-name	The name (if available) of the procedure that owns the stack frame
owner-lb	The value of the LB (linkage base) register belonging to the procedure that owns the stack frame

pcl-addr	Address of the PCL instruction that caused the procedure to be invoked
return-addr	The address to which the procedure returns

Notes

A called-from or returns-to value such as 0(0) or 0(0)/177776 usually means that the stack frame has an invalid return point and can never return. An example of such a frame is the first frame set up by SEG in a V-mode static mode program.

If the stack switches to a different segment during the trace, DUMP_STACK displays the following message:

```
STACK SEGMENT IS n
```

where **n** is the octal segment number of the new stack segment.

Format of a Fault Frame

If the frame is a fault frame, the following format is used:

(nn) offset: FAULT FRAME; fault type = fault-type.
Fault returns to ret-pb; LB= faulter-lb, keys= faulter-keys.
Fault code= f-code; fault addr= f-addr.
Registers at time of fault:
Save Mask=ssssss; XB= xb-value
000001 000002 000003 000004 000005 000006
000007 000010 000011 000012 000013 000014
000015 000016 000017 000020 000021 000022
000023 000024 000025 000026 000027 000030

Parameter	*Definition*
fault-type	Location in the fault table of the type of fault that occurred
ret-pb	Address to which the fault returns
faulter-lb	LB register belonging to the procedure in which the fault occurred
faulter-keys	CPU keys at the time of the fault
register data	If present, a direct dump of the register save area (in the same format as that produced by the CPU RSAV instruction)
f-code	Fault code generated by this particular fault
f-addr	Fault address generated by this particular fault

Format of a Condition Frame

If the activation is a condition frame, the following format is used:

```
(nn) offset: CONDITION FRAME for "condition-name"; returns to ret-pb.
  Condition raised at sigloc; LB= siglb; keys= sigkeys.
  [(Crawlout to outerpb; LB= outerlb; keys= outerkeys.)]
  [Registers at time of fault in inner ring:
      Save Mask=ssssss; XB= xb-value
      000001 000002 000003 000004 000005 000006
      000007 000010 000011 000012 000013 000014
      000015 000016 000017 000020 000021 000022
      000023 000024 000025 000026 000027 000030]
```

The latter two items are displayed only if the condition was signaled in an inner ring and if subsequently a crawlout to the current ring occurred.

C Old Commands

Introduction

This appendix describes the following commands:

- AMLC, which was made obsolete by the SET_ASYNC command
- ASRCWD, which is needed only by a user with a serial printer, a serial card reader, or a serial card punch
- The elaborate form of ATTACH, which became obsolete when pathnames were allowed in commands
- FUTIL, which has been replaced by the COPY, DELETE, LD, RWLOCK, and SIZE commands
- PROTEC, which was made obsolete by the PROTECT command

▶ AMLC [protocol] line [config_number [lword]]

Operator Command

AMLC configures an asynchronous line connected to an AMLC or ICS controller. The AMLC command is obsolete; its functional replacement is the SET_ASYNC command. For further information, see the *System Administrator's Guide, Volume II: Communication Lines and Controllers*.

See also SET_ASYNC.

▶ ASRCWD number

ASRCWD sets a virtual control word that selects the input or output device for diverted terminal I/O.

ASRCWD functions only on a system that has a serial asynchronous interface with serial printers, serial card readers, and/or serial card punches. After the command is given, input is taken from and output is sent to the two devices selected by bits 11 through 16 of the octal value **number**.

INPUT BITS (xx):	00	User Terminal
	01	(Reserved)
	10	(Reserved)
	11	Serial Card Reader
OUTPUT BITS (yyyy):	0000	User Terminal
	1000	User Terminal
	0100	Serial Printer 1
	0010	Serial Printer 2
	0001	Serial Card Punch

For example, to choose the serial card reader for input and the user terminal for output, select the following bits:

```
xx yyyy

11 0000
```

which can be regrouped as 110 000 to become octal 60. The command would be

```
OK, ASRCWD 60
```

The virtual control word is usually set to the appropriate value by programs. ASRCWD is used only when a program exits abnormally (for example, by pressing the BREAK or CONTROL-P keys), leaving input or output diverted away from the terminal. The command

```
ASRCWD 0
```

corrects this condition.

ATTACH directory [password] [ldisk] [key]

ATTACH finds the disk file location of **directory**, checks **password** (if any), and places this information into two storage areas associated with the user. These two areas define the current directory and the home directory. Each area contains the name of the directory, its disk location, and a status flag that indicates whether the user is an owner or a nonowner of the directory. As an option, the user can specify that the information be recorded to redefine only the current directory, leaving the home directory information unchanged.

ATTACH operates only on file directories, not segment directories.

Passwords

A directory can have a pair of passwords to provide security. Only users who know the owner password have owner access to the files in the directory. The nonowner password likewise restricts nonowner access. See the PASSWD and PROTECT commands in Chapter 2 of this guide.

If a directory has both owner and nonowner passwords, a correct password — owner or nonowner — is required, and the user obtains owner or nonowner status appropriately. An incorrect or missing password results in the NO UFD ATTACHED status.

If a directory has only an owner password, then the correct password gives the user owner status. An incorrect password, or none, gives nonowner status.

If a directory has no password, owner status is given to an attaching user, whether the user supplies a password or not.

Logical Disk

The user can specify which logical disk is to be searched for the directory by using the command format

ATTACH directory password ldisk

where **ldisk** is an octal integer identifying the logical disk. The value of **ldisk** is specified as follows:

Value	*Meaning*
n	Searches the MFD of logical disk **n**. The LDEV column of the STATUS DISKS printout shows the logical disk number for each disk. **n** can be from 0 to the maximum number of disks allowed.
100000	Searches MFDs of all disks in order of increasing **ldisk** number. (Default)
177777	Searches the MFD of the disk to which the user is currently attached.

You can accomplish the same actions by using pathnames. For example,

ATTACH <n>pathname

ATTACH ordinary-pathname

ATTACH <*>pathname

correspond exactly to **disk** values of **n**, 100000, and 177777, respectively.

Home Key Values

ATTACH normally sets both the current and the home directories to the target directory. However, in the command format

ATTACH directory password ldisk key

you can choose a value of **key** to allow or prevent the setting of the home directory to be **directory**. Specify **key** as follows:

Key	*Meaning*
177777	Attaches to a directory in the MFD on **ldisk** but does not set it as the home directory.
0	Attaches to a directory in the MFD on **ldisk** and sets it as the home directory. (Default)
1	Attaches to a subdirectory in the current directory but does not set it as the home directory.
2	Attaches to a subdirectory in the current directory and sets it as the home directory.

To specify a key without specifying **ldisk**, use the slash convention, as in the following three formats:

ATTACH directory password 1/key

ATTACH directory 1/key

ATTACH pathname 1/key

▶ FUTIL [–NORM]

FUTIL invokes a file utility whose subcommands allow the user to copy, delete, and list both files and directories. After you give the FUTIL command, the utility is loaded and its right angle-bracket prompt (>) is displayed. Enter the FUTIL subcommands at this prompt. You can run FUTIL from a command file. FUTIL has been made obsolete by the COPY, DELETE, LD, and SIZE commands.

For a description of FUTIL, see the *Operator's Guide to System Commands.*

▶ PROTEC pathname key-1 key-2

PROTEC sets access rights for files in a password directory. These rights are for an owner of the file and for nonowners. To set these rights, you must have owner access to the file. **pathname** specifies the name of the file to be protected. **key-1** is an integer that specifies the owner's access rights to **pathname**. **key-2** is an integer that specifies the nonowner's access rights to **pathname**.

The values and meanings for **key-1** and **key-2** are as follows:

Value	*Rights*
0	No access of any kind allowed
1	Read only
2	Write only
3	Read and write
4	Delete and truncate
5	Delete, truncate, and read
6	Delete, truncate, and write
7	All access

Here are two examples of the PROTEC command.

Example 1:

```
OK, PROTEC JULY.MEMO 7 1
```

In this example the owner is given all access rights to JULY.MEMO and read-only rights to nonowners.

Example 2:

```
OK, PROTEC SALES_REPORT 7 7
```

In this example both owners and nonowners receive all access rights to the file SALES_REPORT.

Note

The default protection keys associated with any newly created file or directory are 7 and 0. (Owner is given all rights and nonowner is given none.) If you issue PROTEC without arguments, values are set to 0 and 0. This means the owner has no rights and therefore cannot access the file unless you issue another PROTEC command with an appropriate value for **key-1**.

D ECS & EBCDIC Character Sets

As of Rev. 21.0, Prime has expanded its character set. The basic character set remains the same: it is the ANSI ASCII 7-bit set (called ASCII-7), with the 8th bit turned on. However, the 8th bit is now significant; when it is turned off, it signifies a different character. Thus, the size of the character set has doubled, from 128 to 256 characters. This expanded character set is called the Prime Extended Character Set (Prime ECS).

The pre-Rev. 21.0 character set is a proper subset of Prime ECS. These characters have not changed. Software written before Rev. 21.0 will continue to run exactly as it did before. Software written at Rev. 21.0 that does not use the new characters needs no special coding to use the old ones.

Prime ECS support is automatic at Rev. 21.0. You may begin to use characters that have the 8th bit turned off. However, the extra characters are not available on most printers and terminals. Check with your System Administrator to find out whether you can take advantage of the new characters in Prime ECS.

Table D-1 shows the complete Prime Extended Character Set. The pre-Rev. 21.0 character set consists of the characters with decimal values 128 through 255 (octal values 200 through 377). (The pre-Rev. 21.0 character set is shaded in Table D-1.) The characters added at Rev. 21.0 all have decimal values less than 128 (octal values less than 200).

Specifying Prime ECS Characters

Direct Entry

On terminals that support Prime ECS, you can enter the characters directly. For information on how to do this, see the appropriate manual for your terminal.

Octal Notation

If you use the Editor (ED), you can enter any Prime ECS character on any terminal by typing:

```
^octal-value
```

where *octal-value* is the three-digit octal number given in Table D-1. You must type all three digits, including leading zeros. You can specify characters in this way anywhere in a program.

Character String Notation

The way in which you specify Prime ECS characters in character strings in programs depends on the character that you wish to specify. Note that you can specify Prime ECS characters on any terminal by using one of the notations shown below. However, the characters themselves can only appear on a terminal that supports Prime ECS. A terminal supports Prime ECS if:

- It uses ASCII-8 as its internal character set, and
- The TTY8 protocol is configured on your asynchronous line

If you do not know whether your terminal supports Prime ECS, ask your System Administrator.

For rules describing how to specify Prime ECS characters in character strings, refer to the appropriate language manual.

Special Meanings of Prime ECS Characters

Either PRIMOS or an applications program running on PRIMOS may interpret some Prime ECS characters in a special way. For example, PRIMOS interprets ^P as a process interrupt. ED, the Editor, interprets the backslash (\) as a logical tab. If you wish to make use of the Prime ECS backslash character in a file you are editing with ED, you must define another character as your logical tab.

For a detailed description of how PRIMOS interprets the following Prime ECS characters, see the discussion in the *PRIMOS User's Guide* of special terminal keys and special characters: ^‘ \‘ "‘ ?‘ ^P‘ ^S‘ ^Q‘ _‘ and ;.

Prime Extended Character Set Table

Table D-1 contains all of the Prime ECS characters, arranged in ascending order. This order provides both the collating sequence and the way that comparisons are

done for character strings. For each character, the table includes the graphic, the mnemonic, the description, and the binary, decimal, hexadecimal, and octal codes. A blank entry indicates that the particular item does not apply to this character. The graphics for control characters are specified as **^character**; for example, ^P represents the character produced when you type P while holding the Control key down.

Characters with decimal values from 000 to 031 and from 128 to 159 are control characters.

Characters with decimal values from 032 to 127 and from 160 to 255 are graphic characters.

TABLE D-1
The Prime Extended Character Set

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
	RES1	Reserved for future standardization	0000 0000	000	00	000
	RES2	Reserved for future standardization	0000 0001	001	01	001
	RES3	Reserved for future standardization	0000 0010	002	02	002
	RES4	Reserved for future standardization	0000 0011	003	03	003
	IND	Index	0000 0100	004	04	004
	NEL	Next line	0000 0101	005	05	005
	SSA	Start of selected area	0000 0110	006	06	006
	ESA	End of selected area	0000 0111	007	07	007
	HTS	Horizontal tabulation set	0000 1000	008	08	010
	HTJ	Horizontal tab with justify	0000 1001	009	09	011
	VTS	Vertical tabulation set	0000 1010	010	0A	012
	PLD	Partial line down	0000 1011	011	0B	013
	PLU	Partial line up	0000 1100	012	0C	014
	RI	Reverse index	0000 1101	013	0D	015
	SS2	Single shift 2	0000 1110	014	0E	016
	SS3	Single shift 3	0000 1111	015	0F	017
	DCS	Device control string	0001 0000	016	10	020
	PU1	Private use 1	0001 0001	017	11	021
	PU2	Private use 2	0001 0010	018	12	022
	STS	Set transmission state	0001 0011	019	13	023
	CCH	Cancel character	0001 0100	020	14	024
	MW	Message waiting	0001 0101	021	15	025
	SPA	Start of protected area	0001 0110	022	16	026
	EPA	End of protected area	0001 0111	023	17	027
	RES5	Reserved for future standardization	0001 1000	024	18	030
	RES6	Reserved for future standardization	0001 1001	025	19	031
	RES7	Reserved for future standardization	0001 1010	026	1A	032
	CSI	Control sequence introducer	0001 1011	027	1B	033
	ST	String terminator	0001 1100	028	1C	034
	OSC	Operating system command	0001 1101	029	1D	035
	PM	Privacy message	0001 1110	030	1E	036

TABLE D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
	APC	Application program command	0001 1111	031	1F	037
	NBSP	No-break space	0010 0000	032	20	040
¡	INVE	Inverted exclamation mark	0010 0001	033	21	041
¢	CENT	Cent sign	0010 0010	034	22	042
£	PND	Pound sign	0010 0011	035	23	043
¤	CURR	Currency sign	0010 0100	036	24	044
¥	YEN	Yen sign	0010 0101	037	25	045
¦	BBAR	Broken bar	0010 0110	038	26	046
§	SECT	Section sign	0010 0111	039	27	047
¨	DIA	Diaeresis, umlaut	0010 1000	040	28	050
©	COPY	Copyright sign	0010 1001	041	29	051
ª	FOI	Feminine ordinal indicator	0010 1010	042	2A	052
«	LAQM	Left angle quotation mark	0010 1011	043	2B	053
¬	NOT	Not sign	0010 1100	044	2C	054
	SHY	Soft hyphen	0010 1101	045	2D	055
®	TM	Registered trademark sign	0010 1110	046	2E	056
¯	MACN	Macron	0010 1111	047	2F	057
°	DEGR	Degree sign	0011 0000	048	30	060
±	PLMI	Plus/minus sign	0011 0001	049	31	061
²	SPS2	Superscript two	0011 0010	050	32	062
³	SPS3	Superscript three	0011 0011	051	33	063
´	AAC	Acute accent	0011 0100	052	34	064
µ	LCMU	Lowercase Greek letter µ, micro sign	0011 0101	053	35	065
¶	PARA	Paragraph sign, Pilgrow sign	0011 0110	054	36	066
·	MIDD	Middle dot	0011 0111	055	37	067
¸	CED	Cedilla	0011 1000	056	38	070
¹	SPS1	Superscript one	0011 1001	057	39	071
º	MOI	Masculine ordinal indicator	0011 1010	058	3A	072
»	RAQM	Right angle quotation mark	0011 1011	059	3B	073
¼	FR14	Common fraction one-quarter	0011 1100	060	3C	074

TABLE D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
½	FR12	Common fraction one-half	0011 1101	061	3D	075
¾	FR34	Common fraction three-quarters	0011 1110	062	3E	076
¿	INVQ	Inverted question mark	0011 1111	063	3F	077
À	UCAG	Uppercase A with grave accent	0100 0000	064	40	100
Á	UCAA	Uppercase A with acute accent	0100 0001	065	41	101
Â	UCAC	Uppercase A with circumflex	0100 0010	066	42	102
Ã	UCAT	Uppercase A with tilde	0100 0011	067	43	103
Ä	UCAD	Uppercase A with diaeresis	0100 0100	068	44	104
Å	UCAR	Uppercase A with ring above	0100 0101	069	45	105
Æ	UCAE	Uppercase diphthong Æ	0100 0110	070	46	106
Ç	UCCC	Uppercase C with cedilla	0100 0111	071	47	107
È	UCEG	Uppercase E with grave accent	0100 1000	072	48	110
É	UCEA	Uppercase E with acute accent	0100 1001	073	49	111
Ê	UCEC	Uppercase E with circumflex	0100 1010	074	4A	112
Ë	UCED	Uppercase E with diaeresis	0100 1011	075	4B	113
Ì	UCIG	Uppercase I with grave accent	0100 1100	076	4C	114
Í	UCIA	Uppercase I with acute accent	0100 1101	077	4D	115
Î	UCIC	Uppercase I with circumflex	0100 1110	078	4E	116
Ï	UCID	Uppercase I with diaeresis	0100 1111	079	4F	117
Ð	UETH	Uppercase Icelandic letter Eth	0101 0000	080	50	120
Ñ	UCNT	Uppercase N with tilde	0101 0001	081	51	121
Ò	UCOG	Uppercase O with grave accent	0101 0010	082	52	122
Ó	UCOA	Uppercase O with acute accent	0101 0011	083	53	123

TABLE. D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
Ô	UCOC	Uppercase O with circumflex	0101 0100	084	54	124
Õ	UCOT	Uppercase O with tilde	0101 0101	085	55	125
Ö	UCOD	Uppercase O with diaeresis	0101 0110	086	56	126
×	MULT	Multiplication sign used in mathematics	0101 0111	087	57	127
Ø	UCOO	Uppercase O with oblique line	0101 1000	088	58	130
Ù	UCUG	Uppercase U with grave accent	0101 1001	089	59	131
Ú	UCUA	Uppercase U with acute accent	0101 1010	090	5A	132
Û	UCUC	Uppercase U with circumflex	0101 1011	091	5B	133
Ü	UCUD	Uppercase U with diaeresis	0101 1100	092	5C	134
Ý	UCYA	Uppercase Y with acute accent	0101 1101	093	5D	135
Þ	UTHN	Uppercase Icelandic letter Thorn	0101 1110	094	5E	136
ß	LGSS	Lowercase German letter double s	0101 1111	095	5F	137
à	LCAG	Lowercase a with grave accent	0110 0000	096	60	140
á	LCAA	Lowercase a with acute accent	0110 0001	097	61	141
â	LCAC	Lowercase a with circumflex	0110 0010	098	62	142
ã	LCAT	Lowercase a with tilde	0110 0011	099	63	143
ä	LCAD	Lowercase a with diaeresis	0110 0100	100	64	144
å	LCAR	Lowercase a with ring above	0110 0101	101	65	145
æ	LCAE	Lowercase diphthong ae	0110 0110	102	66	146
ç	LCCC	Lowercase c with cedilla	0110 0111	103	67	147
è	LCEG	Lowercase e with grave accent	0110 1000	104	68	150
é	LCEA	Lowercase e with acute accent	0110 1001	105	69	151
ê	LCEC	Lowercase e with circumflex	0110 1010	106	6A	152

TABLE D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
ë	LCED	Lowercase e with diaeresis	0110 1011	107	6B	153
ì	LCIG	Lowercase i with grave accent	0110 1100	108	6C	154
í	LCIA	Lowercase i with acute accent	0110 1101	109	6D	155
î	LCIC	Lowercase i with circumflex	0110 1110	110	6E	156
ï	LCID	Lowercase i with diaeresis	0110 1111	111	6F	157
ð	LETH	Lowercase Icelandic letter Eth	0111 0000	112	70	160
ñ	LCNT	Lowercase n with tilde	0111 0001	113	71	161
ò	LCOG	Lowercase o with grave accent	0111 0010	114	72	162
ó	LCOA	Lowercase o with acute accent	0111 0011	115	73	163
ô	LCOC	Lowercase o with circumflex	0111 0100	116	74	164
õ	LCOT	Lowercase o with tilde	0111 0101	117	75	165
ö	LCOD	Lowercase o with diaeresis	0111 0110	118	76	166
÷	DIV	Division sign used in mathematics	0111 0111	119	77	167
ø	LCOO	Lowercase o with oblique line	0111 1000	120	78	170
ù	LCUG	Lowercase u with grave accent	0111 1001	121	79	171
ú	LCUA	Lowercase u with acute accent	0111 1010	122	7A	172
û	LCUC	Lowercase u with circumflex	0111 1011	123	7B	173
ü	LCUD	Lowercase u with diaeresis	0111 1100	124	7C	174
ý	LCYA	Lowercase y with acute accent	0111 1101	125	7D	175
þ	LTHN	Lowercase Icelandic letter Thorn	0111 1110	126	7E	176
ÿ	LCYD	Lowercase y with diaeresis	0111 1111	127	7F	177

TABLE D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
	NUL	Null	1000 0000	128	80	200
^A	SOH/TC1	Start of heading	1000 0001	129	81	201
^B	STX/TC2	Start of text	1000 0010	130	82	202
^C	ETX/TC3	End of text	1000 0011	131	83	203
^D	EOT/TC4	End of transmission	1000 0100	132	84	204
^E	ENQ/TC5	Enquiry	1000 0101	133	85	205
^F	ACK/TC6	Acknowledge	1000 0110	134	86	206
^G	BEL	Bell	1000 0111	135	87	207
^H	BS/FE0	Backspace	1000 1000	136	88	210
^I	HT/FE1	Horizontal tab	1000 1001	137	89	211
^J	LF/NL/FE2	Line feed	1000 1010	138	8A	212
^K	VT/FE3	Vertical tab	1000 1011	139	8B	213
^L	FF/FE4	Form feed	1000 1100	140	8C	214
^M	CR/FE5	Carriage return	1000 1101	141	8D	215
^N	SO/LS1	Shift out	1000 1110	142	8E	216
^O	SI/LS0	Shift in	1000 1111	143	8F	217
^P	DLE/TC7	Data link escape	1001 0000	144	90	220
^Q	DC1/XON	Device control 1	1001 0001	145	91	221
^R	DC2	Device control 2	1001 0010	146	92	222
^S	DC3/XOFF	Device control 3	1001 0011	147	93	223
^T	DC4	Device control 4	1001 0100	148	94	224
^U	NAK/TC8	Negative acknowledge	1001 0101	149	95	225
^V	SYN/TC9	Synchronous idle	1001 0110	150	96	226
^W	ETB/TC10	End of transmission block	1001 0111	151	97	227
^X	CAN	Cancel	1001 1000	152	98	230
^Y	EM	End of medium	1001 1001	153	99	231
^Z	SUB	Substitute	1001 1010	154	9A	232
^[	ESC	Escape	1001 1011	155	9B	233
^\	FS/IS4	File separator	1001 1100	156	9C	234
^]	GS/IS3	Group separator	1001 1101	157	9D	235
^^	RS/IS2	Record separator	1001 1110	158	9E	236
^_	US/IS1	Unit separator	1001 1111	159	9F	237
	SP	Space	1010 0000	160	A0	240
!		Exclamation mark	1010 0001	161	A1	241
"		Quotation mark	1010 0010	162	A2	242
#	NUMB	Number sign	1010 0011	163	A3	243
$	DOLR	Dollar sign	1010 0100	164	A4	244
%		Percent sign	1010 0101	165	A5	245
&		Ampersand	1010 0110	166	A6	246

TABLE D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
'		Apostrophe	1010 0111	167	A7	247
(		Left parenthesis	1010 1000	168	A8	250
)		Right parenthesis	1010 1001	169	A9	251
*		Asterisk	1010 1010	170	AA	252
+		Plus sign	1010 1011	171	AB	253
,		Comma	1010 1100	172	AC	254
–		Minus sign	1010 1101	173	AD	255
.		Period	1010 1110	174	AE	256
/		Slash	1010 1111	175	AF	257
0		Zero	1011 0000	176	B0	260
1		One	1011 0001	177	B1	261
2		Two	1011 0010	178	B2	262
3		Three	1011 0011	179	B3	263
4		Four	1011 0100	180	B4	264
5		Five	1011 0101	181	B5	265
6		Six	1011 0110	182	B6	266
7		Seven	1011 0111	183	B7	267
8		Eight	1011 1000	184	B8	270
9		Nine	1011 1001	185	B9	271
:		Colon	1011 1010	186	BA	272
;		Semicolon	1011 1011	187	BB	273
<		Less than sign	1011 1100	188	BC	274
=		Equal sign	1011 1101	189	BD	275
>		Greater than sign	1011 1110	190	BE	276
?		Question mark	1011 1111	191	BF	277
@	AT	Commercial at sign	1100 0000	192	C0	300
A		Uppercase A	1100 0001	193	C1	301
B		Uppercase B	1100 0010	194	C2	302
C		Uppercase C	1100 0011	195	C3	303
D		Uppercase D	1100 0100	196	C4	304
E		Uppercase E	1100 0101	197	C5	305
F		Uppercase F	1100 0110	198	C6	306
G		Uppercase G	1100 0111	199	C7	307
H		Uppercase H	1100 1000	200	C8	310
I		Uppercase I	1100 1001	201	C9	311
J		Uppercase J	1100 1010	202	CA	312
K		Uppercase K	1100 1011	203	CB	313
L		Uppercase L	1100 1100	204	CC	314
M		Uppercase M	1100 1101	205	CD	315
N		Uppercase N	1100 1110	206	CE	316

TABLE D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
O		Uppercase O	1100 1111	207	CF	317
P		Uppercase P	1101 0000	208	D0	320
Q		Uppercase Q	1101 0001	209	D1	321
R		Uppercase R	1101 0010	210	D2	322
S		Uppercase S	1101 0011	211	D3	323
T		Uppercase T	1101 0100	212	D4	324
U		Uppercase U	1101 0101	213	D5	325
V		Uppercase V	1101 0110	214	D6	326
W		Uppercase W	1101 0111	215	D7	327
X		Uppercase X	1101 1000	216	D8	330
Y		Uppercase Y	1101 1001	217	D9	331
Z		Uppercase Z	1101 1010	218	DA	332
[	LBKT	Left bracket	1101 1011	219	DB	333
\	REVS	Reverse slash, backslash	1101 1100	220	DC	334
]	RBKT	Right bracket	1101 1101	221	DD	335
^	CFLX	Circumflex	1101 1110	222	DE	336
_		Underline, underscore	1101 1111	223	DF	337
`	GRAV	Left single quote, grave accent	1110 0000	224	E0	340
a		Lowercase a	1110 0001	225	E1	341
b		Lowercase b	1110 0010	226	E2	342
c		Lowercase c	1110 0011	227	E3	343
d		Lowercase d	1110 0100	228	E4	344
e		Lowercase e	1110 0101	229	E5	345
f		Lowercase f	1110 0110	230	E6	346
g		Lowercase g	1110 0111	231	E7	347
h		Lowercase h	1110 1000	232	E8	350
i		Lowercase i	1110 1001	233	E9	351
j		Lowercase j	1110 1010	234	EA	352
k		Lowercase k	1110 1011	235	EB	353
l		Lowercase l	1110 1100	236	EC	354
m		Lowercase m	1110 1101	237	ED	355
n		Lowercase n	1110 1110	238	EE	356
o		Lowercase o	1110 1111	239	EF	357
p		Lowercase p	1111 0000	240	F0	360
q		Lowercase q	1111 0001	241	F1	361
r		Lowercase r	1111 0010	242	F2	362
s		Lowercase s	1111 0011	243	F3	363
t		Lowercase t	1111 0100	244	F4	364

TABLE D-1
The Prime Extended Character Set - Continued

Graphic	Mnemonic	Description	Binary	Decimal	Hex	Octal
u		Lowercase u	1111 0101	245	F5	365
v		Lowercase v	1111 0110	246	F6	366
w		Lowercase w	1111 0111	247	F7	367
x		Lowercase x	1111 1000	248	F8	370
y		Lowercase y	1111 1001	249	F9	371
z		Lowercase z	1111 1010	250	FA	372
{	LBCE	Left brace	1111 1011	251	FB	373
|	VERT	Vertical line	1111 1100	252	FC	374
}	RBCE	Right brace	1111 1101	253	FD	375
~	TIL	Tilde	1111 1110	254	FE	376
	DEL	Delete	1111 1111	255	FF	377

TABLE D-2
EBCDIC Character Set

Decimal	*Octal*	*Hex.*	*Char.*	*Decimal*	*Octal*	*Hex.*	*Char.*
000	000	00	NUL	048	060	30	
001	001	01	SOH	049	061	31	
002	002	02	STX	050	062	32	SYN
003	003	03	ETX	051	063	33	
004	004	04	PF	052	064	34	PN
005	005	05	HT	053	065	35	RS
006	006	06	LC	054	066	36	UC
007	007	07	DEL	055	067	37	EOT
008	010	08		056	070	38	
009	011	09		057	071	39	
010	012	0A	SMM	058	072	3A	
011	013	0B	VT	059	073	3B	CU3
012	014	0C	FF	060	074	3C	DC4
013	015	0D	CR	061	075	3D	NAK
014	016	0E	SO	062	076	3E	
015	017	0F	SI	063	077	3F	SUB
016	020	10	DLE	064	100	40	Sp
017	021	11	DC1	065	101	41	
018	022	12	DC2	066	102	42	
019	023	13	TM	067	103	43	
020	024	14	RES	068	104	44	
021	025	15	NL	069	105	45	
022	026	16	BS	070	106	46	
023	027	17	IL	071	107	47	
024	030	18	CAN	072	110	48	
025	031	19	EM	073	111	49	
026	032	1A	CC	074	112	4A	
027	033	1B	CU1	075	113	4B	.
028	034	1C	IFS	076	114	4C	<
029	035	1D	IGS	077	115	4D	(
030	036	1E	IRS	078	116	4E	+
031	037	1F	IUS	079	117	4F	
032	040	20	DS	080	120	50	&
033	041	21	SOS	081	121	51	
034	042	22	FS	082	122	52	
035	043	23		083	123	53	
036	044	24	BYP	084	124	54	
037	045	25	LF	085	125	55	
038	046	26	ETB	086	126	56	
039	047	27	ESC	087	127	57	
040	050	28		088	130	58	
041	051	29		089	131	59	
042	052	2A	SM	090	132	5A	!
043	053	2B	CU2	091	133	5B	$
044	054	2C		092	134	5C	*
045	055	2D	ENQ	093	135	5D	)
046	056	2E	ACK	094	136	5E	;
047	057	2F	BEL	095	137	5F	

TABLE D-2
EBCDIC Character Set - Continued

Decimal	*Octal*	*Hex.*	*Char.*	*Decimal*	*Octal*	*Hex.*	*Char.*
096	140	60	-	143	217	8F	
097	141	61	/	144	220	90	
098	142	62		145	221	91	j
099	143	63		146	222	92	k
100	144	64		147	223	93	l
101	145	65		148	224	94	m
102	146	66		149	225	95	n
103	147	67		150	226	96	o
104	150	68		151	227	97	p
105	151	69		152	230	98	q
106	152	6A	|	153	231	99	r
107	153	6B	,	154	232	9A	
108	154	6C	%	155	233	9B	
109	155	6D	_	156	234	9C	
110	156	6E	>	157	235	9D	
111	157	6F	?	158	236	9E	
112	160	70		159	237	9F	
113	161	71		160	240	A0	
114	162	72		161	241	A1	~
115	163	73		162	242	A2	s
116	164	74		163	243	A3	t
117	165	75		164	244	A4	u
118	166	76		165	245	A5	v
119	167	77		166	246	A6	w
120	170	78		167	247	A7	x
121	171	79	‘	168	250	A8	y
122	172	7A	:	169	251	A9	z
123	173	7B	#	170	252	AA	
124	174	7C	@	171	253	AB	
125	175	7D	’	172	254	AC	
126	176	7E	=	173	255	AD	
127	177	7F	"	174	256	AE	
128	200	80		175	257	AF	
129	201	81	a	176	260	B0	
130	202	82	b	177	261	B1	
131	203	83	c	178	262	B2	
132	204	84	d	179	263	B3	
133	205	85	e	180	264	B4	
134	206	86	f	181	265	B5	
135	207	87	g	182	266	B6	
136	210	88	h	183	267	B7	
137	211	89	i	184	270	B8	
138	212	8A		185	271	B9	
139	213	8B		186	272	BA	
140	214	8C		187	273	BB	
141	215	8D		188	274	BC	
142	216	8E		189	275	BD	

TABLE D-2
EBCDIC Character Set - Continued

Decimal	*Octal*	*Hex.*	*Char.*	*Decimal*	*Octal*	*Hex.*	*Char.*
190	276	BE		223	337	DF	
191	277	BF		224	340	E0	\
192	300	C0	{	225	341	E1	
193	301	C1	A	226	342	E2	S
194	302	C2	B	227	343	E3	T
195	303	C3	C	228	344	E4	U
196	304	C4	D	229	345	E5	V
197	305	C5	E	230	346	E6	W
198	306	C6	F	231	347	E7	X
199	307	C7	G	232	350	E8	Y
200	310	C8	H	233	351	E9	Z
201	311	C9	I	234	352	EA	
202	312	CA		235	353	EB	
203	313	CB		236	354	EC	(bank)
204	314	CC	(bank)	237	355	ED	
205	315	CD		238	356	EE	
206	316	CE	(bank)	239	357	EF	
207	317	CF		240	360	F0	0
208	320	D0	}	241	361	F1	1
209	321	D1	J	242	362	F2	2
210	322	D2	K	243	363	F3	3
211	323	D3	L	244	364	F4	4
212	324	D4	M	245	365	F5	5
213	325	D5	N	246	366	F6	6
214	326	D6	O	247	367	F7	7
215	327	D7	P	248	370	F8	8
216	330	D8	Q	249	371	F9	9
217	331	D9	R	250	372	FA	|
218	332	DA		251	373	FB	
219	333	DB		252	374	FC	
220	334	DC		253	375	FD	
221	335	DD		254	376	FE	
222	336	DE		255	377	FF	

E ECL Commands By Function

This appendix divides the ECL commands into functional groups. An example illustrates the use of each command. The series of examples forms a terminal session, with each command acting upon the result of the previous command line. The cursor position is denoted by an underscore (_). At the beginning of the session, you are examining the contents of the BAR subdirectory. However, you have mistyped the string **BAR** on the command line. At this point, the command line looks like this:

```
LD *>ABR>@@_
```

Now go on to the following sections for a running example of the uses of ECL.

Cursor Movement

This section illustrates the use of the cursor movement commands.

Command	*Keystroke*	*Result*	*Comment*
begin_line	Ctrl A	`LD *>ABR>@@`	
end_line	Ctrl E	`LD *>ABR>@@_`	
back_char	Esc 7 Ctrl B	`LD *>ABR>@@`	(Executed 7 times)
forward_char	Ctrl F	`LD *>ABR>@@`	
forward_word	Esc F	`LD *>ABR>@@`	
back_word	Esc B	`LD *>ABR>@@`	

Insertion

ECL causes all printable characters to insert themselves into the command line. Non-printable characters can also be inserted.

Command	*Syntax*	*Keystroke*	*Result*
self_insert		A B	`LD *>ABABR>@@`
quote	Esc Q	Esc Q Ctrl C	`LD *>AB^CABR>@@`
do_quote	Esc ' Esc '	Esc ' Ctrl D	`LD *>AB^C^DABR>@@`

Deletion

You can delete by characters, words, lines, or regions.

Command	*Keystroke*	*Result*	*Comment*
delete_char	Ctrl D	`LD *>AB^C^DBR>@@`	
rubout_char	Ctrl H	`LD *>AB^CBR>@@`	(PRIMOS erase)
delete_word	Esc D	`LD *>AB^C>@@`	
rubout_word	Esc Ctrl H	`LD *>>@@`	
kill_line	Ctrl K	`LD *>`	
kill_region	Ctrl U Ctrl W		(Kills line)

Deletion Recall

All deletion commands except *delete_char* and *rubout_char* save the deleted text in the kill ring. Most consecutive deletions are kept together in the same kill ring entry.

Command	*Keystroke*	*Result*	*Comment*
yank	Ctrl Y	`LD *>_`	(From *kill_region* above)
	Ctrl Y	`LD *> LD *>_`	
yank_replace	Esc Y	`LD *>AB^CBR>@@_`	(From *delete_word*, *rubout_word*, and *kill_line* above)

Regions

This section introduces the ECL commands that work specifically with regions. A *region* is a user-defined portion of text. The beginning of the region is called the *mark*, which is invisible on the terminal screen; the end of the region is called the *point*. (Often, the point is where the cursor rests.) In the examples used in this section, the mark is denoted by the red underscore.

Command	*Keystroke*	*Result*	*Comment*
copy_region	Esc W	`LD *>AB^CBR>@@_`	
exchange_mark	Ctrl X Ctrl X	`LD *>AB^CBR>@@_`	
kill_region	Ctrl W	`LD *>`	
yank	Ctrl Y	`LD *>AB^CBR>@@_`	(From *kill_region*)
yank_replace	Esc Y	`LD *>AB^CBR>@@_`	(From *copy_region*)
mark	Ctrl @	`LD *>AB^CBR>@@`	

Repetition

This section introduces the ECL commands that allow you to reexecute commands or to execute commands any number of times.

Command	*Syntax*	*Keystroke*	*Result*
multiplier	Ctrl U *[digit(s)]* *command*	Ctrl U Ctrl B	`LD *>AB^CBR>@@`
		Ctrl U 2 X	`LD *>AB^CBXXR>@@`
esc_digit			
	Esc *[digit(s)]* *command*	Esc 2 Ctrl H	`LD *>AB^CBR>@@`
reexecute		Ctrl C	`LD *>ABR>@@`

Twiddle

This section illustrates the character position-switching ECL command.

Command	*Keystroke*	*Result*
twiddle	Ctrl T	`LD *>BAR>@@`

The desired command line is ready for submittal, now that your session of experimenting with various ECL commands has been completed.

Submit and Abort

This section illustrates the ECL commands that allow you to submit a command to ECL and to abort an ECL command in progress.

Command	*Keystroke*	*Comment*
submit	Ctrl J	(Typed as carriage return)
abort_cmd	Ctrl G	(Aborts operation)

Case Conversion

This section illustrates the case conversion ECL commands. Another simple example begins:

Command	*Keystroke*	*Result*
		fILE.Bin
toggle_case	Ctrl Ctrl	**FILE.Bin**
downcase_word	Esc L	**File.Bin**
upcase_word	Esc U	**File.BIN**

Pathname and Abbreviation Expansion

Pathname expansion is described in Chapter 6. There are basically three types in addition to the menu-selection capability:

Command	*Keystroke*	*Comment*
expand_wild	Ctrl I	(Completes if possible)
	Ctrl U Ctrl I	(Displays matches)
	Ctrl U Ctrl U Ctrl I	(Expands verbatim to full pathname)
expand_wild_menu	Esc *number* Esc Ctrl I	(Selects menu-list entry *number*)

You can expand an abbreviation four ways:

expand_abbrev	Esc A	(Abbrev/gvar-expand single token)
	Esc 0 Esc A	(Abbrev-only-expand single token)
	Ctrl U Esc A	(Abbrev-only-expand entire command line)
	Ctrl U Ctrl U Esc A	(Abbrev/gvar expand entire command line)

History and Refresh

This section illustrates the ECL commands that allow you to manipulate the command history and to refresh the display.

Command	*Keystroke*	*Comment*
refresh	Ctrl L	(Performs display cleanup)
	Esc *number* Ctrl L	(Displays last *number* commands)
prev_line	Ctrl Z	(Recalls last command)
next_line	Ctrl N	(Moves forward in history)
goto_line	Esc *number* Esc G	(Recalls command *number*)
reverse_search	Esc R	(Searches back through history)
forward_search	Esc S	(Searches forward through history)

Keyboard Macros

This section lists the ECL commands that allow you to begin and end a macro, and to execute a macro.

Command	*Keystroke*	*Comment*
collect_macro	Esc (	(Starts keystroke collection)
finish_macro	Esc)	(Finishes keystroke collection)
execute_macro	Esc E	(Executes collected keystrokes)

Binding and PFkey Support

These commands support the ability to change the default command bindings and provide flexibility in programming PF commands.

Command	*Keystroke*	*Comment*
explain_key	Ctrl _	(Reveals key sequence binding)
extend_command	Esc X	(Specifies command by name)
pfkey1 - pfkey512		(Programmable function keys)
do_echo	Esc @ *character(s)* Esc @	(Echos *character(s)* to terminal)
echo_raw		(Echos key sequence back to terminal)
unbound		(Rings the bell for key sequence)

Miscellaneous

The commands in this section allow you, respectively, to invisibly enter a password and to display the version of ECL on your system.

Command	*Keystroke*	*Comment*
password	Esc P	(Invisibly enters password)
version	Ctrl V	(Displays package version)

Index

Symbols

| character, 6-9
" " characters, 1-7
character, 2-89
$$ command, 2-123, 2-302
% character, 1-8, 2-3, 3-2, 4-2
( ) characters, 1-7 to 1-8, 4-3
* command, 2-302
+ character, 1-8, 4-5, 4-21
- character, 1-6
... characters, 1-7
/* command, 2-303
; character, 1-8, 4-2, 5-4
< command, 6-3, 6-7
< > characters, 1-7
= character, 4-20
> character, 2-93
@ character, 1-8, 4-5, 6-4
[ ] characters, 1-7 to 1-8, 3-1
^ character, 1-8, 4-5, 4-22, 6-2, D-3
~ character, 1-8, 2-8, 2-93, 4-23, 5-4, 6-2 to 6-3, 6-33

A

A command, 2-1, 2-21, C-2
ABBREV command, 2-1
ABBREV function, 3-2
Abbreviation expansion, 2-93
Abbreviation file
  activating, 2-6
  creating, 2-1 to 2-2
  deactivating, 2-6
  returning status of, 3-2
Abbreviation preprocessor, 2-1
Abbreviation Verify mode, 2-8
Abbreviations
  defining, 2-2
  deleting, 2-7
  expansion of by PRIMOS, 5-3
  in CPL programs, 2-1
  listing, 2-7
  procedures for use of, 2-1
  use in multiuser environment, 2-6
  user-defined versus PRIMOS-defined, 2-8
  variables in, 2-3
abortcmd, 6-31
Aborting batch jobs, 2-124
Aborting file transfer requests, 2-113
Absolute pathname, definition of, 1-2
Accepting messages, 2-182
Access categories
  changing name of, 2-42
  copying, 2-56
  creating, 2-84, 2-236 to 2-237
  definition of, 2-234
  deleting, 2-74
  listing contents of, 2-142
  listing size of, 2-252
  modifying, 2-84
Access Control Lists
  see: ACLs
Access groups
  adding to existing ACLs, 2-84
  definition of, 2-233
  deleting from ACLs, 2-85
  listing, 2-150
Access rights
  changing segment, 2-250
  list of, 2-233
  setting, 2-232
  setting in password directories, 2-211
  setting on files with PROTEC command, C-4
–ACCESS_CATEGORY wildcard option, 4-8
Accessing the system, 2-169
ACL directories, using PROTECT in, 2-212
ACLs
  creating, 2-84, 2-236
  editing, 2-84
  listing contents of, 2-142
  listing priority, 2-155
  removing priority, 2-217
  setting, 2-232
  setting devices with, 2-79
  setting priority, 2-244
  specific, 2-234
Activating global variable files, 2-72
Activating ROAM files, 2-218
Adding components to generated names, 4-21
Adding disks
  with ADDISK command, 2-10
  with DISKS command, 2-80
ADDISK command, 2-10
ADD_REMOTE_ID command, 2-10
ADMIN_LOG command, 2-11
AFTER function, 3-3
–AFTER wildcard option, 4-8
Altered commands, summary of, 1-27
AMLC command, C-1
AMLC commands, converting, 2-55
Angle brackets < > in pathnames, 1-7
ARCHIVE command, 2-12
ARCHIVE tapes
  listing catalogs of, 2-144
  listing contents of, 2-167
ARCHIVE_RELEASE command, 2-12
ARCHIVE_RESTORE command, 2-13
Argument abbreviations, 2-5
Arguments in command lines, 1-5

ARID command, 2-10
Arithmetic command functions, summary of, 1-25
ASRCWD command, 2-13, C-1
Assembler, invoking, 2-205
ASSIGN command, 2-14
Assignable Disks Table, 2-19, 2-80
Assigning asynchronous lines, 2-19
Assigning disks, 2-19
Assigning lines, 2-20
Assigning magnetic tapes, 2-15
Assigning NTS lines, 2-20
Assigning peripheral devices, 2-14
Assigning plotters, 2-213
Assigning SMLC lines, 2-20
Assigning user number to supervisor terminal, 2-297
Asterisk (*) command, 2-302
Asterisk in pathnames, 1-3
Asynchronous lines
  assigning, 2-19
  characteristics of, 2-239
  configuring, 2-12, 2-238
  default settings for, 2-243
  displaying status of, 2-144
  setting characteristics of, 2-287
  unassigning, 2-292
At sign @ in wildcarding, 1-8, 4-5
ATM command, 2-20
ATM_ADMIN command, 2-20
ATTACH command, 2-21, C-2
Attaching to another directory, 2-21
Attaching to origin directory, 2-199
ATTRIB function, 3-4
Attribute files, 2-266
Attributes
  definition of, 2-266
  user, 2-96
Attributes of ROAM files
  listing, 2-246
  setting, 2-246
Audit Collection facility, 2-230
Audit files, 2-208
  performing backups on, 2-290
Audit trail, 2-208
Automatic pathname completion, 2-93, 6-3
AVAIL command, 2-23

B

B command, 2-25
-BACKEDUP_AFTER wildcard option, 4-8
-BACKEDUP_BEFORE wildcard option, 4-8
backchar, 6-10, 6-32
Backslash character (card option), 6-9
BACKUP command, 2-25
BACKUP tapes
  listing catalogs of, 2-144
  listing contents of, 2-167
BACKUP_RELEASE command, 2-26
BACKUP_RESTORE command, 2-26
backword, 6-32
Basename, definition of, 1-4
BASIC command, 2-26
BASIC interpreter, 2-26, 2-71
BASIC programs
  executing, 2-26, 2-71
  renumbering, 2-195
BASICV command, 2-27
BATCH command, 2-28
Batch jobs
  aborting, 2-124
  canceling, 2-124
  information on, 2-27
  managing submitted, 2-123
  modifying, 2-124
  monitoring, 2-124
  storing submittal options, 2-302
  submitting, 2-120
Batch monitor, passing parameters to, 2-302
Batch queue information, 2-28
BATGEN command, 2-29
BEFORE function, 3-5
-BEFORE wildcard option, 4-8
beginline, 6-8, 6-32
BINARY command, 2-31
Binary editor, 2-83
Binary output files, opening, 2-31
BIND command, 2-32
Bindings
  finding current, 6-25
  reestablishing fundamental, 6-23
Bindings file
  compiling, 6-23
  definition of, 6-8
  PST 100, 6-16
  PT200, 6-16
  PT45, 6-21
  restoring, 6-24
  sample, 6-15
  saving, 6-23
  saving and restoring simultaneously, 6-24
  syntax of, 6-8
Bindings, key
  see: Key bindings in ECL
Boot tapes, creating, 2-32
BOOT_CREATE command, 2-32
-BOTTOM_UP treewalking option, 4-17
Braces { } in command formats, 1-7
Brackets, angle, 1-7
Brackets, square, 1-7 to 1-8, 3-1
BREAK key, enabling or disabling, 2-287, 6-2
Breaks in ring, finding, 2-107
Bringing up a PRIME MEDUSA workstation, 2-179
Building concatenated files
  with CONCAT command, 2-49
  with RJQ command, 2-225
Building MIDASPLUS files, 2-127

C

C command, 2-32
C compiler, 2-34
CAB command, 2-33
CAD/CAM commands, summary of, 1-23
CALC function, 3-5
Canceling batch jobs, 2-124
Canceling file transfer requests, 2-113
Canceling queued plot files, 2-204
Canceling queued RJE files, 2-224
Canceling spooled files, 2-276
Capital letters, use in command formats, 1-6
Card punch
  assigning, 2-14
  punching files onto, 2-68
Card reader
  assigning, 2-14
  reading cards from, 2-70
Caret (^), 6-9
  use in name generation, 4-22
  use in wildcarding, 4-5
  use on command line, 1-8
Case in command line, 1-6
Catalogs, tape, 2-144
Categoryname, definition of, 1-4
CBL command, 2-33
CBLDML command, 2-33
CBLSUBS command, 2-34
CC command, 2-34
Chaining COMI files, 2-45
CHANGE_PASSWORD command, 2-35
Changes in commands, summary of, 1-26
Changing ACL directories to password directories, 2-221
Changing ACLs, 2-84
Changing characteristics of terminals, 2-286
Changing directory names, 2-42
Changing execution priority level, 2-36
Changing filenames, 2-42
Changing login password, 2-35
Changing message receive state, 2-182
Changing names of file system objects, 2-42
Changing ROAM filenames, 2-43
Changing system prompts, 2-213
Changing to another directory, 2-21
Changing to origin directory, 2-199
CHAP command, 2-36
Checking message receive states, 2-183
Checking storage status of disks, 2-23
Cleaning out MIDASPLUS files, 2-127
Clearing user address space, 2-105
CLOSE command, 2-39
Closing COMI files, 2-44
Closing COMO files, 2-47
Closing files
  summary of commands, 1-13
  with CLOSE command, 2-39
CLUP command, 2-40
CMDNC0 directory, 1-9
CMPF command, 2-40
CNAME command, 2-42

CND_INFO function, 3-6
CN_RBF command, 2-43
CO command, 2-43
COBOL 74 compiler, 2-33
COBOL 74 DDL compiler, 2-33
COBOL 74 DML preprocessor, 2-33
collectmacro, 6-32
Combining command-line features, 4-22
Combining files
  with CONCAT command, 2-49
  with MRGF command, 2-186
  with SORT command, 2-260
COMI files
  $$ command in, 2-123
  as phantoms, 2-201
  chaining, 2-45
  closing, 2-44
  executing, 2-44
  pausing, 2-44
  used to submit batch jobs, 2-120
  using, 2-45
COMINPUT command, 2-43
Command abbreviations, defining, 2-4
Command directory (CMDNC0), 1-9
Command environment
  listing attributes of, 2-153
  resetting, 2-117
  summary of commands to define, 1-16
Command execution, summary of commands to display information on, 1-18
Command functions
  evaluation of by PRIMOS, 5-4
  summary of, 1-25
  supression of, 2-8
Command history, 2-92, 6-34
Command input files
  see: COMI files
Command levels
  releasing, 2-215
  status information on, 2-214
Command line
  arguments on, 1-5
  entrynames in, 1-5
  format of, 1-5
  iteration in, 4-3
  length of, 1-6
  list of special characters, 1-8
  multiple commands on, 4-3
  order of processing on, 5-1
  pathnames in, 1-5
  special characters in, 1-8
  suppressing special characters on, 4-23
Command-line editing, 6-1
Command-line editor (ECL), 2-86
Command output files
  see: COMO files
Command processor, operation of, 1-9
Command separator character, 1-8, 4-2
Command-line features
  combining, 4-22
  list of, 4-1
  use with COPY command, 2-64
  use with DELETE command, 2-75
Commands
  at mini-command level, 2-154
  deferring execution of, 2-130
  issuing multiple, 4-3
  online information on, 2-116
  order of execution by PRIMOS, 5-1
  with iteration lists, 4-3
COMM_CONTROLLER command, 2-46
Comment lines, designating, 2-303
Communicating with other users, 2-180
Communications commands, summary of, 1-20
COMO command, 2-46
COMO files
  closing, 2-47
  dating, 2-71
  opening, 2-46
  pausing, 2-47
  using, 2-48
COMOUTPUT command, 2-46
Comparing files
  with CMPF command, 2-40
  with FILVER command, 2-106
Compilers
  CBL (COBOL 74), 2-33
  CBLSUBS (COBOL 74 DDL), 2-34
  CC (C), 2-34
  F77 (FORTRAN 77), 2-103
  F77SUBS (FORTRAN 77 DDL), 2-104
  FSUBS (FORTRAN IV DDL), 2-108
  FTN (FORTRAN IV), 2-109
  PASCAL, 2-200
  PL1G, 2-204
  summary of, 1-14
  VRPG (V mode), 2-301
Compute time, displaying, 2-289
CONCAT command, 2-49
Concatenating files
  with CONCAT command, 2-49
  with RJQ command, 2-225
Concurrency locks on files, setting, 2-226
Condition on the stack, 3-6
CONFIG command, 2-53
CONFIG_DSM command, 2-54
CONFIG_NET command, 2-54
CONFIG_UM command, 2-54
Configuration preprocessor, 2-53
Configurator program, 2-255
Configuring a PRIME MEDUSA installation, 2-179
Configuring asynchronous lines, 2-12, 2-238
Connect time, displaying, 2-289
Connecting to other networked systems, 2-191
Control characters in ECL, 6-1
Controllers
  displaying information on communication, 2-144
  shutting down, 2-46
CONTROL-P, enabling or disabling, 2-287, 6-2
CONTROL-Q, enabling, 2-90, 2-288
CONTROL-S, enabling, 2-90, 2-288
Conventions, explanation of, 1-6
CONVERT_AMLC_COMMANDS utility, 2-55
CONVERT_ENV utility, 2-55
Converting ACL directories to password directories, 2-221
Converting ASCII files to uppercase, 2-294
Converting decimal integers to hexadecimal, 3-17
Converting decimal integers to octal, 3-18
Converting hexadecimal numbers to decimal, 3-10
Converting octal numbers to decimal, 3-12
COPY command, 2-55
Copy operations, incremental, 2-61
COPY_DISK command, 2-67
Copying access categories, 2-56
Copying disks to other disks, 2-67
Copying file system objects, 2-55
Copying files, 2-55
Copying files across networks, 2-112
Copying ROAM files, 2-67
Copying to an open target file, 2-59
Copying with password directories, 2-66
COPY_RBF command, 2-67
copyregion, 6-32
CPL command, 2-68
CPL programs
  executing with CPL command, 2-68
  executing with RESUME command, 2-220
  restarting, 2-278
  running as phantoms, 2-202
  use for submitting batch jobs, 2-120
  use of SET_VAR command in, 2-250
  user-defined abbreviations in, 2-1
CPMPC command, 2-68
CPU time, displaying, 2-289
CPU time usage, 2-214
CPW command, 2-35
CRASH_AUDIT command, 2-69
CREATE command, 2-69
Creating boot tapes, 2-32
Creating directories, 2-69
Creating EPF runfiles, 2-32
Creating global variable files, 2-72
Creating global variables, 2-249
Creating segment directories, 2-230
Creating user abbreviations, 2-2
CREATK command, 2-70
CRMPC command, 2-70
Cross-product iteration lists, 4-4
Current directory, 2-131, 2-199
Current environment, commands for information on, 1-17

D

Dash, 1-6
Data management commands, summary of, 1-21
Data Terminal Ready signal, dropping, 2-81
Date
  displaying current calendar, 2-71
  setting system, 2-231
DATE command, 2-71
DATE function, 3-7
DBACP command, 2-71
DBASIC command, 2-71
DBG command, 2-72
DBMS commands, summary of, 1-21
DBMS schema DDL compiler, 2-229
DBMS Schema Decompiler (SCHDEC), 2-229
DBMS Schema Editor (SCHED), 2-229
DBUTL command, 2-72
Deactivating global variable files, 2-73
Debuggers
  DBG (Source Level), 2-72
  HPSD, 2-116
  PSD, 2-119
  PSD (R-mode version), 2-213
  summary of, 1-15
  VPSD, 2-301
  VPSD16, 2-301
Debugging obsolete programs, 2-206
Decimal integers
  converting to hexadecimal, 3-17
  converting to octal, 3-18
Decompiler, schema, 2-229
Default system prompts, changing, 2-213
Deferring execution of commands, 2-130
Deferring messages, 2-182
Deferring phantom logout notification, 2-175
DEFINE_GVAR command, 2-73
Defining global variables, 2-249
Defining user abbreviations, 2-2
DELAY command, 2-74
Delaying execution of commands, 2-130
DELETE command, 2-74
deletechar, 6-32
Delete protection on file system objects, 2-243
DELETE_RBF command, 2-77
DELETE_VAR command, 2-77
deleteword, 6-32
Deleting components in generated names, 4-22
Deleting global variable files, 2-73
Deleting global variables, 2-77
Deleting MIDASPLUS files, 2-127
Deleting ROAM files, 2-77
Deleting user-defined abbreviations, 2-7
Deleting users from ACLs, 2-85
DELSEG command, 2-78
Designating comment lines, 2-303
Device name, definition of, 1-2
DEVICE_ACLS command, 2-79
Devices
  assigning, 2-14
  displaying all assigned, 2-143
  setting ACLs on, 2-79
  status information on, 2-282
  unassigning, 2-292
DIAG command, 2-79
Diagnostic Utility (DIAG), 2-79
DIR function, 3-7
Directories
  attaching to, 2-21
  changing passwords in, 2-200
  commands for protecting, 1-12
  converting to password, 2-221
  copying, 2-56
  creating, 2-69
  deleting, 2-74
  listing access rights, 2-142
  listing contents, 2-131
  listing quotas, 2-156
  listing size of, 2-252
  origin, 2-199
  replacing, 2-59
  setting access rights, 2-142
  setting delete protection on, 2-243
  setting quotas on, 2-245
  summary of commands, 1-12
  using ATTRIB function on, 3-4
  using the DIR function on, 3-7
  using treewalking in, 4-12
–DIRECTORY wildcard option, 4-8
Directoryname, definition of, 1-4
Disabling PRIMENET, 2-285
Discarding stack history, 2-215
Discarding terminal buffers, 2-226
Disconnecting from lines, 2-81
DISCOVER command, 2-79
Disk I/O time, displaying, 2-289
Diskname, definition of, 1-2
Disks
  adding with ADDISK command, 2-10
  adding with DISKS command, 2-80
  assigning, 2-19
  copying with COPY_DISK utility, 2-67
  definition of, 1-2
  displaying status information of, 2-146
  formatting, 2-178
  listing priority ACL, 2-155
  removing priority ACL, 2-217
  repairing, 2-108
  setting priority ACL on, 2-244
  shutting down, 2-251
  status information on, 2-282
  unassigning, 2-293
DISKS command, 2-80
Display manager, 6-27
Displaying computer time used, 2-289
Displaying contents of files, 2-254
Displaying current date and time, 2-71
Displaying directory contents, 2-131
Displaying files on terminals, 2-254
Displaying the number of users, 2-297
DISPLAY_LOG command, 2-80
Distributed System Management
  see: DSM
DISTRIBUTE_DSM command, 2-80
Diverted terminal I/O, 2-13, C-1
Dividing numbers with MOD function, 3-11
DMSTK command, 2-82, B-1
Documents, online list of Prime, 2-116
doecho, 6-2, 6-11, 6-32
doquote, 6-2, 6-11, 6-14 to 6-15, 6-25, 6-32
Double-precision BASIC interpreter, 2-71
downcaseword, 6-32
DPTCFG command, 2-80
DPTX command, 2-81
DPTX queues, information on, 2-81
DPTX/DSC commands
  DPTCFG, 2-80
  DPTX, 2-81
  DPTXMTR, 2-81
  OWLDSC, 2-199
  PRTDSC, 2-212
  PT45DSC, 2-213
  TCF, 2-286
DPTXMTR command, 2-81
DPTX/TCF facility, 2-286
DROPDTR command, 2-81
Dropping the DTR signal, 2-81
DSM
  administering logs, 2-11
  copying master configuration files, 2-80
  creating and modifying configuration files, 2-54
  displaying disk information with, 2-146
  displaying file unit status with, 2-168
  displaying LAN information with, 2-151
  displaying logs from, 2-80
  displaying memory usage with, 2-154
  displaying port assignments with, 2-155
  displaying PRIMENET links with, 2-155
  displaying PRIMENET remote nodes with, 2-155
  displaying semaphore values with, 2-162
  displaying synchronous line configuration with, 2-167
  displaying user process characteristics of, 2-156
  displaying virtual circuits with, 2-168
  event message handling, 2-54
  initializing, 2-280
  listing communications controllers with, 2-144
  status of, 2-284
  stopping, 2-284
  writing logs from, 2-80
DTAR2 private segments, information on, 2-160
DTR signal, dropping, 2-81
Dumping MIDASPLUS files, 2-179
DUMP_SEGMENT command, 2-82

DUMP_STACK command, 2-82, 2-205 to 2-206, B-1
DUMP_USER command, 2-82
Duplex, full or half, 2-287

E

E command, 2-101
echoraw, 6-11, 6-33
ECL
  automatic pathname completion in, 6-3
  basic options, 2-88
  command history, 2-92 to 2-93
  command line options, 2-86
  command output (COMO) files, 6-3
  commands reference, 6-31
  comment character, 6-9
  completing pathnames with, 6-3
  control characters, 6-1
  controlling terminal I/O, 2-91
  display manager, 6-27
  editing characteristics, 2-94
  examples, 2-94
  introduction, 2-86, 6-1
  invoked as EPF command function, 6-3
  key bindings, 6-8
  kill ring, 6-2
  pathname completion, 2-93
  programmable function commands, 6-12
  prompt handling, 6-28
  replacing pathnames with, 6-4
  search ring, 6-2
  terminal bindings and characteristics, 2-89 to 2-90
  TERMINFO capabilities, 6-29
  treewalk wildcard expansion, using, 6-4
  using commands of, 6-2
ECL, automatic pathname completion in, 6-3
ECL command, 2-86
ED command, 2-83
EDAC command, 2-84
EDB command, 2-83
EDIT_ACCESS command, 2-84
EDIT_CMD_LINE command, 2-86
Editing ACLs, 2-84
Editors
  ED (line), 2-83
  EDB (binary), 2-83
  EMACS (screen), 2-101
  FED (FORMS), 2-105
  NSED (nonshared ED), 2-193
  PRIMOS profile, 2-96
  SCHED (schema), 2-229
EDIT_PROFILE command, 2-96
Eligibility timeslice, modifying, 2-101
ELIGTS command, 2-101
Ellipsis ... in command formats, 1-7
EMACS command, 2-101
Emptying terminal buffers, 2-226
endline, 6-33
ENTRYNAME function, 3-8
Entrynames
  definition of, 1-3
  suffixes of, 1-4
  use in command lines, 1-5
Entrypoint search lists, listing contents of, 2-158
Entrypoint search rules, expanding objectname to fully qualified pathname, 2-101
Entrypoints, listing library EPF, 2-151
Environment commands, printer, 2-210
EPF command function, ECL invoked as, 6-3
EPF commands
  INITIALIZE_COMMAND_ENVIRONMENT, 2-117
  LIST_EPF, 2-146
  LIST_LIBRARY_ENTRIES, 2-151
  LIST_LIMITS, 2-153
  LIST_MINI_COMMANDS, 2-154
  LIST_SEARCH_RULES, 2-158
  LIST_SEGMENT, 2-160
  REMOVE_EPF, 2-216
EPFs
  creating, 2-32
  displaying information on, 2-146
  executing, 2-219
  listing entrypoints, 2-151
  output, 2-147
  removing from address space, 2-216
  REPLACE files, 2-65
  replacing mapped, 2-65
  restarting, 2-278
Equal sign (=), use in name generation, 4-20
ER! prompt, changing, 2-214
Erase character, 2-287
Erasing files, 2-74
Erasing ROAM files, 2-77
Error prompt, changing, 2-214
ERRSET subroutine, 2-206
ERRVEC, displaying contents of, 2-206
escdigit, 6-2, 6-31, 6-33, 6-35
Establishing your identity, commands for, 1-11
Evaluating expressions, 3-5
exchangemark, 6-33
executemacro, 6-12 to 6-13, 6-32 to 6-33
Executing BASIC programs, 2-26, 2-71
Executing COMI files, 2-44
Executing CPL programs, 2-68, 2-220
Executing EPFs, 2-219
Executing PRIME MEDUSA utility programs, 2-179
Executing R-mode runfiles, 2-220
Executing SEG runfiles, 2-230
Execution priority level, 2-36
EXISTS function, 3-8
expandabbrev, 6-33
Expanding abbreviations, 5-3
Expanding partial pathnames, 6-5
expandwild, 2-89, 2-93, 6-3 to 6-5, 6-33
expandwildmenu, 6-6, 6-34
explainkey, 6-11, 6-25, 6-34
extendcommand, 6-34
External command, definition of, 1-9

F

F77 command, 2-103
F77DML command, 2-103
F77SUBS command, 2-104
Failed Login, 2-170
FAP command, 2-104
FAU command, 2-104
FDML command, 2-105
FED command, 2-105
Fetching files from remote systems, 2-113
File Administrator Utility, invoking, 2-104
File system command functions, summary of, 1-26
File system objects, 1-2
File transfer assistance (online), 2-114
File transfer queues, information on, 2-114
File Transfer Request utility, invoking, 2-109
File Transfer Service
  FTGEN, 2-108
  FTOP, 2-109
  FTR, 2-109
  summary of commands, 2-114
File types, sorting different, 2-264
File-units, status of, 2-283
File utility (FUTIL), C-4
-FILE wildcard option, 4-8
Filename, definition of, 1-4
Files
  archiving ROAM, 2-228
  attribute, 2-266
  building MIDASPLUS, 2-127
  changing names of, 2-43
  closing, 2-39
  COMI (command input), 2-43
  commands for protecting, 1-14
  COMO (command output), 2-46
  comparing, 2-40, 2-106
  concatenating, 2-49
  concatenating RJE, 2-225
  converting to uppercase, 2-294
  copying, 2-55
  copying a master DSM configuration, 2-80
  copying across networks, 2-112, 2-223
  deleting, 2-74
  deleting MIDASPLUS, 2-127
  deleting ROAM, 2-77
  displaying on terminal, 2-254
  dumping MIDASPLUS, 2-179
  fetching from other systems, 2-113
  formatting ASCII, 2-226
  global variable, 2-72, 2-249
  listing, 2-142
  listing attributes of ROAM, 2-157, 2-246
  listing size of, 2-251
  merging, 2-186, 2-256, 2-260
  merging with COPY command, 2-60
  opening, 2-31, 2-119, 2-142, 2-197, 3-12

Files (continued)
printing, 2-266
printing on MPC parallel printer, 2-208
printing on serial interface printers, 2-212
protecting, 1-14, 2-211
reading next line in, 3-12
renaming, 2-43
replacing, 2-59
restoring ROAM, 2-218
setting access rights in password directories, 2-211
setting access rights on, 2-232, C-4
setting delete protection on, 2-243
setting read/write locks, 2-226
size of, 2-251
sorting, 2-255, 2-285
temporary destinations, 2-110
transferring to other systems, 2-109, 2-223
types for input sorting, 2-256
using ATTRIB function on, 3-4
verifying PRISAM, 2-79
writing text with WRITE_FILE function, 3-22
FILMEM command, 2-105
FILVER command, 2-106
FIND_RING_BREAK command, 2-107
finishmacro, 6-32, 6-34
FIX_DISK command, 2-108
Forced logout, 2-174
Force-deleting protected objects, 2-75
Format of PRIMOS command line, 1-5
Formatting ASCII files, 2-226
Formatting disks, 2-178
FORMS Administrative Processor, 2-104
FORMS commands, summary of, 1-22
FORMS editor, invoking, 2-105
FORTRAN 77 compiler, 2-103
FORTRAN 77 DDL compiler, 2-104
FORTRAN 77 DML preprocessor, 2-103
FORTRAN DDL compiler, 2-109
FORTRAN DML preprocessor, 2-105
FORTRAN IV compiler, 2-109
forwardchar, 6-10, 6-34
forwardsearch, 2-94, 6-2, 6-34
forwardword, 2-94, 6-34
FSUBS command, 2-108
FTGEN command, 2-108
FTN command, 2-109
FTOP command, 2-109
FTR command, 2-109
FTS, 2-114
Full pathname, definition of, 1-3
Full-duplex mode, setting, 2-287
Fully qualified pathname, EXPAND_SEARCH_RULES command, provided by, 2-101
Function keys, 6-14
codes for, 6-25
FUTIL command, C-4

G

GENERATE_CATALOG command, 2-115
Generating pathnames, 4-19
Generation patterns, 4-20
GET_VAR function, 3-9
Global variable character, 1-8
Global variable files
activating, 2-73
creating, 2-72
deleting, 2-73
returning pathname of, 3-9
Global variables
creating, 2-249
defining, 2-249
deleting, 2-77
listing, 2-168
returning value of, 3-9
use in CPL programs, 2-250
use of, 4-2
gotoline, 6-2, 6-34
GVPATH function, 3-10

H

Half duplex, setting terminal line to, 2-287
Half-duplex networks
controlling connections of, 2-191
information on, 2-115
Handling terminal I/O, commands for, 1-18
HDX sites, monitoring, 2-115
HDXSTAT command, 2-115
HELP command, 2-116
HEX function, 3-10
Home directory, C-4
HPSD command, 2-116
Hyphen, meaning of, 1-6

I

I command, 2-119
ICE command, 2-117, 2-248
IDBMS command, 2-116
Incorporating shared code into PRIMOS, 2-250
Incremental copy operations, 2-61
INDEX function, 3-10
INFO command, 2-117
Initial Attach Point
assigning, 2-96
at login, 2-169
returning to, 2-199
INITIALIZE_COMMAND_ENVIRONMENT command, 2-117
Initializing ROAM, 2-119
INPUT command, 2-119
Internal command, definition of, 1-9
Interpreters
BASIC, 2-26
BASICV, 2-27
DBASIC, 2-71
Inverted wildcard matching, 4-5
I/O device handling commands, 1-19
I/O time usage, 2-214
I/O usage time, displaying, 2-289
IPSD command, 2-119
IROAM command, 2-119
ISC, server, 2-162
Italicized words, meaning of, 1-7
Iteration
combined with other features, 4-22
explanation of, 4-3
processing by PRIMOS, 5-5
use with DELETE command, 2-75

J

JOB command, 2-119
Job processing commands, summary of, 1-16

K

KBUILD command, 2-127
Key bindings in ECL, 6-8
Keyed-index files, creating or modifying, 2-70
Keys, sort, 2-257
KIDDEL command, 2-127
Kill buffer, 6-35, 6-37
Kill character, setting, 2-287
killline, 6-35
killregion, 2-91, 6-32, 6-35
Kill ring, 2-93, 6-2

L

L command, 2-142
LAB command, 2-128
LABEL command, 2-129
Labels, magnetic tape, 2-129
LAC command, 2-142
LAN Host Controller 300 (LHC), 2-151
LAN Terminal Server 300 (LTS), 2-153
LATE command, 2-130
LCB command, 2-145
LD command, 2-130 to 2-131
LE command, 2-146
Leaving the system, 2-173
LEM command, 2-140
LENGTH function, 3-11
Length of command line, 1-6
LG command, 2-150
Library EPFs, listing entrypoints of, 2-151
Line editors
ED, 2-83
NSED, 2-193
Line printers
assigning, 2-14
printing on, 2-266
recovering output from, 2-13, C-2
LINEFEED, echoing, 2-287
Lines, assigning, 2-20
Linker, BIND, 2-32
Linking EPF runfiles, 2-32

Linking loader
LOAD, 2-169
SEG, 2-230
LIST_ACCESS command, 2-142
LIST_ASSIGNED_DEVICES command, 2-143
LIST_ASYNC command, 2-144
LIST_CATALOG command, 2-144
LIST_COMM_CONTROLLERS command, 2-144
LIST_CONFIG command, 2-144
LIST_CONTIGUOUS_BLOCKS command, 2-145
LIST_DISKS command, 2-146
LIST_DUMP command, 2-146
LIST_EPF command, 2-146
LIST_EXTENT_MAP command, 2-140
LISTF command, 2-142
LIST_GROUP command, 2-150
Listing abbreviations, 2-7
Listing ACL groups, 2-150
Listing ACLs, 2-142
Listing attributes on ROAM files, 2-157, 2-246
LISTING command, 2-142
Listing contents of directories, 2-131
Listing NTS associations, 2-194
Listing priority ACL, 2-155
Listing queued RJE files, 2-224
Listing size of file system objects, 2-251
Listing spooled files, 2-272
LIST_LAN_NODES command, 2-151
LIST_LHC_STATUS command, 2-151
LIST_LIBRARY_ENTRIES command, 2-151
LIST_LIMITS command, 2-153
LIST_LTS_STATUS command, 2-153
LIST_MEMORY command, 2-154
LIST_MINI_COMMANDS command, 2-154
LIST_PRIMENET_LINKS command, 2-154
LIST_PRIMENET_NODES command, 2-155
LIST_PRIMENET_PORTS command, 2-155
LIST_PRIORITY_ACCESS command, 2-155
LIST_PROCESS command, 2-156
LIST_QUOTA command, 2-156
LIST_RBF command, 2-157
LIST_REMOTE_ID command, 2-158
LIST_SEARCH_RULES command, 2-158
LIST_SEGMENT command, 2-160
LIST_SEMAPHORES command, 2-162
LIST_SERVER_NAMES command, 2-162
LIST_SESSIONS command, 2-163
LIST_SYNC command, 2-167
LIST_TAPE command, 2-167
LIST_UNITS command, 2-168
LIST_VAR command, 2-168
LIST_VCS command, 2-168
LL command, 2-153
LLENT command, 2-151
LMC command, 2-154
LO command, 2-173
LOAD command, 2-169
Loader
LOAD, 2-169
SEG, 2-230
Local Area Networks
see: Networks
Locks, setting concurrency, 2-226
Logging out, 2-174
Login
examples, 2-172
failed Login attempts, 2-170
login over, 2-171
remote, 2-171
remote password, 2-171
LOGIN command, 2-169
Login password
changing, 2-35
definition of, 2-169
Login Server
starting up, 2-280 to 2-281
stopping, 2-284
Logout
forced, 2-174
functions of, 2-173
logging another user out, 2-174
LOGOUT command, 2-173
Logout notification, phantom, 2-175
Logs
administering DSM, 2-11
displaying DSM, 2-80
LON command, 2-174
LOOK command, 2-176, 2-199
LOOPBACK command, 2-176
Lowercase letters in command formats, 1-6
LPAC command, 2-155
LQ command, 2-156
LRID command, 2-158
LS command, 2-160
LSR command, 2-158

M

M command, 2-180
MAGNET command, 2-177
Magnetic tape commands
ARCHIVE, 2-12
ARCHIVE_RELEASE, 2-12
ARCHIVE_RESTORE, 2-13
ASSIGN, 2-14
BACKUP, 2-25
BACKUP_RELEASE, 2-26
LABEL, 2-129
LIST_CATALOG, 2-144
LIST_TAPE, 2-167
MAGNET, 2-177
MAGRST, 2-178
MAGSAV, 2-178
PHYRST, 2-178, 2-203
PHYSAV, 2-204
REPLY, 2-218
SETMOD, 2-232
TRANSPORT, 2-290
TRANSPORT_RELEASE, 2-291
TRANSPORT_RESTORE, 2-291
Magnetic Tape commands, GENERATE_CATALOG, 2-115
Magnetic tape drives
assigning, 2-15
information on, 2-282
Magnetic tapes, information on contents, 2-167
MAGRST command, 2-178
MAGSAV command, 2-178
MAKE command, 2-178
Making backup tapes, 2-25
Making new directories, 2-69
Managing directories, commands for, 1-11
Managing file transfers, 2-113
Managing files, commands for, 1-12
Managing ROAM files, commands for, 1-13
Managing submitted batch jobs, 2-123
Manuals, online list of Prime, 2-116
Mapped EPFs, removing, 2-216
mark, 6-35
Maxparald command, 2-179
MAXSCH command, 2-178
MAXUSR command, 2-179
MCLUP command, 2-179
MDUMP command, 2-179
MEDCONFIG command, 2-179
MEDUSA command, 2-179
MEDUTIL command, 2-179
Merging files, 2-186, 2-256
Merging files with COPY command, 2-60
Merging segment directories with COPY command, 2-61
Merging sorted files, 2-260
MESSAGE command, 2-180
Message receive states, 2-182
Messages
from operators, 2-184, 2-218
sending local, 2-181
sending to other users, 2-180
to remote users, 2-181
to the operator, 2-181
Metafiles, plotting, 2-204
Metering system usage, 2-294
MFD, attaching to, 2-21
MIDASPLUS cleanup, 2-186
MIDASPLUS commands, summary of, 1-22
Mini-command level, commands at, 2-154
MIRROR_OFF command, 2-185
MIRROR_ON command, 2-185
Miscellaneous functions, 1-26
MOD function, 3-11
Mode, setting, 2-287
–MODIFIED_AFTER wildcard option, 4-8
–MODIFIED_BEFORE wildcard option, 4-8
Modifying ACLs, 2-84
Modifying batch jobs, 2-124
Modifying file transfer requests, 2-113
Modifying spooled files, 2-275
Modulus, returning, 3-11

Monitoring batch jobs, 2-28, 2-124
Monitoring DPTX queues, 2-81
Monitoring file transfer requests, 2-114
Monitoring system usage, 2-294
MONITOR_NET command, 2-185
Moving files, 2-55
Moving ROAM files, 2-67
MPACK command, 2-186
MPC parallel interface printer, 2-208
MPLUSCLUP command, 2-186
MRGF command, 2-186
MRTRESUME command, 2-190
Multiple command sequences, 6-13
Multiple commands
  issuing, 4-3
  processing, 5-4
  with iteration lists, 4-4
multiplier, 6-2, 6-35

N

Name generation
  adding name components, 4-21
  deleting name components, 4-22
  explanation of, 4-19
  processing by PRIMOS, 5-6
Names of file system objects, 1-2
Negation character, 1-8, 4-22
Nested directory copying, 2-59
NET command, 2-191
NETLINK command, 2-191
Networks
  connecting to other systems, 2-191
  half-duplex, 2-115, 2-191
  local area networks, 2-151, 2-153
  monitoring, 2-185
  starting, 2-281
  status information on, 2-283
  stopping, 2-284 to 2-285
New commands, summary of, 1-27
NEWLINE, echoing, 2-287
nextline, 6-2, 6-35
Nonprinting characters, 6-9
Nonquota directories, information on, 2-156
Nonshared version of EDITOR, 2-193
NONTAG sort, 2-256
–NO_VERIFY wildcard option, 4-8
NSED command, 2-193
NTS
  initializing, 2-281
  status information on, 2-283
  stopping, 2-285
  unassigning lines for, 2-293
NTS associations, listing, 2-194
NTS lines, assigning, 2-20
NTS terminal, placing into Command mode, 2-194
NTS_ASSOCIATE command, 2-193
NTS_LINE command, 2-194
NTS_LIST_ASSOCIATE command, 2-194
NTS_UNASSOCIATE command, 2-194
Null command, 2-302
NULL function, 3-11
Number character (#), 6-9
NUMBER command, 2-195
Numbering BASIC programs, 2-195
Number-sign (#) character, 6-28
Number-sign characters (###), 6-35

O

O command, 2-197
OA_ADMIN command, 2-197
OAS command, 2-197
OAS commands, summary of, 1-23
Objectname, definition of, 1-4
Obsolete commands, summary of, 1-28
OCTAL function, 3-12
Octal numbers, converting, 3-12
Office Automation System (OAS)
  commands for, 1-23
  starting up, 2-197
OK, prompt, changing, 2-214
Online information, displaying, 2-116
OPEN command, 2-197
Open files
  checking for, 2-283
  closing, 2-39
OPEN_FILE function, 3-12
Opening COMO files, 2-46
Opening files
  summary of commands for, 1-13
  with BINARY command, 2-31
  with INPUT command, 2-119
  with LISTING command, 2-142
  with OPEN command, 2-197
  with OPEN_FILE function, 3-12
Operator commands, summary of, 1-24
Operator, sending messages to, 2-181
Operator use of MESSAGE command, 2-184
OPRPRI command, 2-199
OR command, 2-199
Order of command processing, 5-1
ORIGIN command, 2-199
Origin directory
  logging in to, 2-169
  returning to, 2-199
OWLDSC command, 2-199

P

P command, 2-205
Pack name, definition of, 1-2
Packet Switched Data Network (PSDN), using, 2-191
Packing MIDASPLUS files, 2-186
Paging, changing percentage of activity of, 2-206
Paper tape files, 2-83
Parentheses ( )
  use for iteration, 1-8, 4-3
  use in command formats, 1-7
Partial pathname, expanding, 6-5
Partial tape dump
  displaying current settings of, 2-146
  resetting parameters, 2-218
Partition name, definition of, 1-2
PASCAL command, 2-200
Passing parameters to batch monitor, 2-302
PASSWD command, 2-200
password, 6-35
Password directories
  attaching to, 2-22
  converting to ACL directories, 2-55
  creating, 2-201
PASSWORD_DIRS command, 2-201
Passwords
  change, 2-170
  changing in directories, 2-200
  changing login, 2-35
  definition of login, 2-169
  expiration, 2-170
Pathname completion, 6-3
PATHNAME function, 3-13
Pathnames
  absolute, 1-2
  definition of, 1-2
  full, 1-3
  fully qualified, with EX_S_R, 2-101
  in command lines, 1-5
  relative, 1-3
Pausing COMI files, 2-44
Pausing COMO files, 2-47
Percent sign (%)
  use in abbreviations, 2-3
  use on command line, 1-8
  use with variables, 4-2
Peripheral devices, assigning, 2-14
PF commands
  PF command, 6-36
  introduction, 6-12
  multiple command sequences, 6-13
  using, 6-12
Pfkey1 - pfkey512, 6-36
PH command, 2-201
PHANTOM command, 2-201
Phantoms
  controlling system printers, 2-208
  logging out, 2-174, 2-203
  logout notification, 2-174
  monitoring, 2-203
  operation of, 2-202
  receiving logout messages, 2-175
  slave, 2-10
  spooler, 2-208
PHYRST command, 2-203
PHYSAV command, 2-204
PL1G command, 2-204
Placing quotation marks around text, 3-14
PLOT command, 2-204
Plot despoolers, managing, 2-204
Plot files, 2-204
Plot queue, 2-204
PLOTOP command, 2-204

Plotters
  assigning, 2-213
  printing files on, 2-204
Plotters, printing files on, 2-212
Plotting files, 2-266
Plus sign (+)
  use in name generation, 4-21
  use in wildcarding, 4-5
  use on command line, 1-8
PM command, 2-205
PMA command, 2-205
Port selector, disconnecting from, 2-81
PRATIO command, 2-206
PRERR command, 2-206
prevline, 6-2, 6-10, 6-36
PRIMAN command, 2-207
Prime INFORMATION, invoking, 2-117
Prime Local Area Networks (LAN), 2-151
Prime Macro Assembler (PMA), 2-205
PRIME MEDUSA commands, summary of, 1-23
Prime RPG II V-mode compiler, 2-301
Prime Symbolic Debugger (PSD), 2-119, 2-213, 2-301
PRIMENET
  displaying all remote nodes of, 2-155
  displaying link status of, 2-154
  displaying port assignments of, 2-155
PRIMENET commands
  CONFIG_NET, 2-54
  FIND_RING_BREAK, 2-107
  FTGEN, 2-108
  FTOP, 2-109
  FTR, 2-109
  HDXSTAT, 2-115
  MONITOR_NET, 2-185
  NET, 2-191
  NETLINK, 2-191
  START_NET, 2-281
  STOP_NET, 2-285
PRIMENET global configuration file, 2-54
PRIME/SNA commands, summary of, 2-207
PRIME/SNA Server Configurator program, 2-255
PRIMEWORD Word Processing system, 2-302
PRIMON command, 2-208
PRIMOS
  shutting down, 2-251
  version of, 2-283
PRIMOS command, 2-208
PRIMOS commands at mini-command level, 2-154
PRIMOS null command, 2-302
Printer attributes, 2-266
Printer emulation program, 2-255
Printer environment commands, 2-210
Printers
  assigning, 2-14
  commands for, 1-20
  displaying information on, 2-208
  status of, 2-209
Printing characters, 6-8
Printing files, 2-266
Printing plot files, 2-204, 2-212, 2-266
Priority ACLs
  listing, 2-155
  removing, 2-217
  setting, 2-244
Priority level, execution, 2-36
PRISAM commands, summary of, 1-22
PRISAM File Administrator Utility, 2-104
PRISAM File Diagnostic Utility, 2-79
PRISAM query language and report writer, 2-79
Private dynamic segments, information on, 2-161
Private static segments, information on, 2-161
PRMPC command, 2-208
Profile editor, 2-96
Profiles, user, 2-96
Programmable function commands
  see: PF commands
Programs
  executing CPL, 2-220
  executing with CPL command, 2-68
  executing with RESUME command, 2-219
  executing with SEG command, 2-230
  linking, 2-32
  loading, 2-169
  loading with SEG command, 2-230
  restarting, 2-278
Project Administrator, 2-96 to 2-97
Project IDs
  assigning, 2-97
  at login, 2-170
  status of, 2-283
Prompt handling, 6-28
Prompts
  SORT, 2-257
  system, 2-213
PROP command, 2-208
PROTEC command, 2-211, C-4
PROTECT command, 2-211
Protecting directories, commands for, 1-12
Protecting files
  summary of commands for, 1-14
  with RWLOCK command, 2-227
  with SET_ACCESS command, 2-232
Protecting objects from deletion, 2-243
PRSER command, 2-212
PRTDSC command, 2-212
PRVER command, 2-212
PSD command, 2-213
PSDN, using, 2-191
PT45DSC command, 2-213
PTELE command, 2-213
Punched cards
  commands for, 1-20
  reading from, 2-70
  writing to, 2-68

Q

QUERY function, 3-14
Queueing files for printing, 2-267
Queueing files via RJQ, 2-223
Queues
  batch, 2-28
  DPTX, 2-81
  FTS, 2-114
  plot, 2-204
  spool, 2-267
Quotas, directory
  listing, 2-156
  setting, 2-245
Quotation marks in text, 1-7
Quotation marks, placing around text, 3-14
quote, 6-12, 6-15, 6-36
QUOTE function, 3-14

R

R command, 2-219
–RBF wildcard option, 4-8
RDY command, 2-213, 2-216
READ_FILE function, 3-15
Reading from punched cards, 2-70
Reading magnetic tape labels, 2-129
Reading next line in a file, 3-15
Read/write concurrency locks, setting, 2-226
Ready prompt, changing, 2-214
Reattaching to origin directory, 2-199
Receive state of terminals
  changing, 2-182
  checking status of, 2-182
Receiving phantom logout messages, 2-175
Recording terminal sessions, 2-46
Recreating ARCHIVE or BACKUP tape catalogs, 2-115
REENTER command, 2-215
Reexecute, 6-36
refresh, 6-35 to 6-36
Registers, contents of, 2-205
Reinitializing stack, 2-215
Reinitializing the default search list, 2-248
Rejecting messages, 2-182
Relative pathname, definition of, 1-3
RELEASE_LEVEL command, 2-215
Releasing command levels, 2-215
Releasing magnetic tapes
  with ARCHIVE_RELEASE command, 2-12
  with BACKUP_RELEASE command, 2-26
REMEPF command, 2-216
Remote disks, adding, 2-10
Remote IDs, deleting, 2-217
Remote Job Entry (RJE) Phase II commands, 2-221
Remote login, 2-171
Remote System User facility, 2-221
Remote user IDs
  adding, 2-11
  listing, 2-158
Remote users, sending messages to, 2-181

REMOVE_EPF command, 2-216
REMOVE_PRIORITY_ACCESS command, 2-217
Removing characters from strings, 3-18
Removing files or directories, 2-74
Removing protection on LOOK command, 2-199
Removing quotation marks from text, 3-15, 3-19
REN command, 2-215
Renaming files or directories, 2-43
Renumbering BASIC programs, 2-195
Repairing disk partitions, 2-108
Replacing file system objects, 2-59
Replacing mapped EPFs, 2-65
Replacing pathnames, 6-4
REPLY command, 2-218
Request numbers, 2-267
Requesting file transfers, 2-110
RESCAN function, 3-15
RESET_DUMP command, 2-218
Resetting queued RJE files, 2-224
Resetting system prompts, 2-213
Resetting user's command environment, 2-117
RESPONSE function, 3-16
$REST identifier
  editing, 2-84
  in priority ACL, 2-244
  setting rights for, 2-233
Restarting EDITOR, 2-83
Restarting programs, 2-278
RESTOR command, 2-218
RESTORE_RBF command, 2-218
Restoring R-mode runfiles to memory, 2-218
Restoring ROAM files, 2-218
Restoring tape data to disks
  from ARCHIVE tapes, 2-13
  from BACKUP tapes, 2-26
  with MAGRST command, 2-178
REST_RBF command, 2-218
Restructuring MIDASPLUS files, 2-186
RESUME command, 2-219
RESUS command, 2-221
Retrieving audit records of memory buffers, 2-69
Returning to origin directory, 2-199
reversesearch, 2-94, 6-2, 6-36
REVERT_PASSWORD command, 2-221
Ring, finding breaks in, 2-107
RJE commands, summary of, 2-221
RJOP command, 2-222
RJQ command, 2-222
RLS command, 2-215
R-mode runfiles, executing, 2-220
ROAM Command Processor Clean-up Program, 2-40
ROAM commands, summary of, 1-13
ROAM files
  activating, 2-218
  changing names of, 2-43
  copying, 2-67
  deleting, 2-77
  listing attributes on, 2-246
  restoring, 2-218
  setting attributes on, 2-246
ROAM System Administrator Utility (ROSAU), 2-225
ROSAU command, 2-225
RPAC command, 2-217
RPG II V-mode compiler, 2-301
RSTERM command, 2-226
ruboutchar, 2-91, 6-36
ruboutword, 2-94, 6-36
Runfiles
  creating EPF (with BIND), 2-32
  creating R-mode (with LOAD), 2-169
  creating SEG, 2-230
  restarting, 2-278
  restoring R-mode, 2-218
Running BASIC programs, 2-26, 2-71
Running COMI files, 2-43
Running CPL programs, 2-68, 2-220
Running CPL programs as phantoms, 2-202
Running PRIME MEDUSA utility programs, 2-179
Running SEG runfiles, 2-230
RUNOFF command, 2-226
RVEC Parameters
  meaning of, A-1
  supplying, A-2
RVEC parameters, displaying, 2-205
RVEC use register, displaying contents of, 2-205
RWLOCK command, 2-226

S

S command, 2-278
SA command, 2-227
SAC command, 2-232
SAVE command, 2-227
SAVE_RBF command, 2-228
Saving disk data to tapes
  with ARCHIVE command, 2-12
  with BACKUP command, 2-25
  with MAGNET command, 2-177
  with MAGSAV command, 2-178
Saving ROAM files to disk, 2-228
SCHDEC command, 2-229
SCHED command, 2-229
SCHEMA command, 2-229
Schema decompiler, 2-229
Schema editor, 2-229
Screen editor, 2-101
SE command, 2-231
SEARCH function, 3-16
Search lists
  resetting to system defaults, 2-248
  setting, 2-248
Search ring, 2-93, 6-2, 6-34
Search rules
  online help, 2-249
  setting, 2-247
Search rules file
  creating, 2-247
  use of, 2-248
SECURITY_MONITOR command, 2-230
SECURITY_STATUS command, 2-230
SEG command, 2-230
Segment 4000g, saving, 2-227
Segment directories
  changing name of, 2-43
  copying, 2-56
  creating, 2-230
  deleting, 2-74
  listing ACLs of, 2-142
  listing size of, 2-252
  protecting, 2-211
  setting access rights on, 2-232
  setting delete protection on, 2-243
-SEGMENT_DIRECTORY wildcard option, 4-8
Segmented programs, 2-230
Segments
  accessing, 2-176
  changing access rights, 2-250
  freeing, 2-78
  information on, 2-160
  memory usage by number of, 2-154
  private dynamic, 2-161
  private static, 2-162
  sharing, 2-250
  writing specific, 2-82
  writing user, 2-82
selfinsert, 6-8, 6-12, 6-14, 6-36
Self-terminating function keys, 6-14
Semaphores, 2-162
  status information on, 2-283
Semicolon as command separator, 1-8, 4-2, 5-4
Sending files, 2-112, 2-223
Sending messages to other users, 2-180
Serial interface printer, printing files on, 2-212
SET_ACCESS command, 2-232
SET_ASYNC command, 2-238
SET_DELETE command, 2-243
SETIME command, 2-231
SETM command, 2-232
SETMOD command, 2-232
SET_PRIORITY_ACCESS command, 2-244
SET_QUOTA command, 2-245
SET_RBF command, 2-246
SET_SEARCH_RULES command, 2-247
SET_TIME_INFO command, 2-249
Setting access rights on password objects, 2-211
Setting attributes on ROAM files, 2-246
Setting delete protection on objects, 2-244
Setting global variables, 2-249
Setting message receive state, 2-182
Setting priority ACLs, 2-244
Setting quotas on directories, 2-245
Setting read/write concurrency locks, 2-226

Setting system date and time, 2-231
Setting terminal characteristics, 1-16, 2-74
Setting terminal line characteristics, 2-238
Setting the erase character, 2-287
Setting the kill character, 2-287
SET_VAR command, 2-249
SH command, 2-251
SHA command, 2-250
SHARE command, 2-250
Sharing segments, 2-250
SHUTDN command, 2-251
Shutting down systems, 2-251
Simple filename, definition of, 1-4
SIZE command, 2-251
Slash (/) character, 2-87
Slash-asterisk command, 2-303
Slave phantom, 2-10
SLIST command, 2-254
SMLC lines, assigning, 2-20
SNA_3270 command, 2-255
SNADSC command, 2-254
SNA_PRINT command, 2-255
SNA_SERVER command, 2-255
SNA_SERVER_CONFIG command, 2-255
SORT command, 2-255
SPAC command, 2-244
Spaces on command line, 1-7
Special characters on command line, 1-8
Specific ACLs, 2-234
SPOOL command, 2-266
Spool queue, listing files in, 2-272
Spooled files
  canceling, 2-276
  listing, 2-272
  modifying, 2-275
Spooler phantoms, status of, 2-209
SPY command, 2-278
SQ command, 2-245
Square brackets [ ]
  use as command function characters, 1-8, 3-1
  use in command formats, 1-7
SSR command, 2-247
Stack
  discarding, 2-215
  dump format of, B-1
  information on, 2-82
START command, 2-278
START_DSM command, 2-280
Starting up disks, 2-10
START_ISCNSR command, 2-280
START_LSR command, 2-281
START_NET command, 2-281
START_NSR command, 2-281
START_NTS command, 2-281
STARTUP command, 2-281
Static program segments, assignment of, 2-231
Static-mode programs, restarting, 2-279
STATUS command, 2-282
Status information on
  abbreviation files, 3-3
  access groups, 2-150
  batch jobs, 2-28, 2-116
  communication controllers, 2-282
  current command level, 2-214
  devices, 2-282
  disks, 2-282
  DPTX queues, 2-81
  EPFs, 2-147
  file units, 2-283
  half-duplex networks, 2-115
  limits of command environments, 2-153
  networks, 2-283
  NTS, 2-283
  project IDs, 2-283
  remote user IDs, 2-158
  segments, 2-160
  semaphores, 2-283
  terminal, 2-287
  user IDs, 2-282
Status of printers, 2-209
STATUS_DSM command, 2-284
STOP_DSM command, 2-284
STOP_LSR command, 2-284
STOP_NET command, 2-285
STOP_NTS command, 2-285
String-handling command functions, summary of, 1-26
Subcommands, EDIT_PROFILE, 2-98
submit, 6-12, 6-14, 6-32, 6-37
Submitting batch jobs, 2-120
Subschema tables
  producing with CBLSUBS command, 2-34
  producing with F77SUBS command, 2-104
  producing with FSUBS command, 2-108
SUBST function, 3-17
SUBSTR function, 3-17
Subsystems, reentering, 2-215
Suffixes, 1-4
Supervisor terminal, assigning user number to, 2-297
Suppressing command-line special characters, 4-23
SVC switch, setting, 2-285
SVCSW command, 2-285
SYNCSORT command, 2-285
Syntax suppression character
  see: Tilde
System date and time, setting, 2-231
System event log, 2-11
System information
  commands, 1-17
  displaying, 2-282
System parameters, defining, 2-53
System printers, displaying information on, 2-208
System prompts, changing, 2-213
System settings commands, summary of, 1-23
System usage, monitoring, 2-294
System users, listing names of, 2-283

T

T command, 2-288
Tab stops, 6-31
TAG sort, 2-256
Tape catalogs, listing, 2-144
Tape commands
  see: Magnetic tape commands
Tape drives, unassigning, 2-293
TCF command, 2-286
TERM command, 2-90, 2-286
Terminal Emulation program, 2-254
Terminal line characteristics, setting, 2-238
Terminal type, specifying, 6-27
Terminals
  deferring execution of commands at, 2-130
  delaying output to, 2-74
  displaying files on, 2-254
  displaying line characteristics of, 2-287
  emptying buffers of, 2-226
  printing text at, 2-292
  recovering from diverted I/O, C-2
  setting delay padding for, 2-74
  setting line characteristics of, 2-286
.TERMINALTYPES, 2-90, 6-23, 6-27
TERMINFO
  database used by, 6-28
  using, 6-30
TERMINFO database, 2-90
Testing for existence of file system objects, 3-8
Text editors
  see: Editors
Text formatting, 2-226
THEMIS command, 2-288
Tilde (~)
  use for syntax suppression, 1-8, 4-23, 5-4
  use in abbreviations, 2-8
Time
  displaying clock, 2-71
  setting system, 2-231
  using date function to return, 3-7
TIME command, 2-288
Timeslice, modifying eligibility, 2-101
togglecase, 6-37
TO_HEX function, 3-17
Tokens in abbreviations, 2-3
TO_OCTAL function, 3-18
TRAMLC command, 2-289
TRANSFER_LOG command, 2-290
Transferring files, 2-110, 2-223, 2-289
TRANSLATE function, 3-18
Translators
  see: Compiler; PMA
TRANSPORT command, 2-290
Transport tapes, listing contents of, 2-167
TRANSPORT_RELEASE command, 2-291
TRANSPORT_RESTORE command, 2-291
Treewalk wildcard expansion, 6-4
Treewalking
  characters, 1-8
  explanation of, 4-12

Treewalking (continued)
implementation of by PRIMOS, 5-5
options, 4-17
Triangle, meaning of, 1-6
TRIM function, 3-18
twiddle, 6-37
TYPE command, 2-291

U

U command, 2-292
UNASSIGN command, 2-292
unbound, 2-89, 6-12, 6-37
Universal time, calculating with SET_TIME_INFO, 2-249
UNQUOTE function, 3-19
UPCASE command, 2-294
upcaseword, 6-37
Uppercase letters
converting ASCII files to, 2-294
use in command formats, 1-6
USAGE command, 2-294
User abbreviations, creating, 2-2
User commands, Summary of, 1-11
User groups
see: Access groups
User IDs
definition of, 2-169
status information on, 2-282
User IDs, remote, 2-11
User, phantom, 2-201
User profiles, 2-96
User segments, 2-160
User-defined abbreviations
see: Abbreviations
USERS command, 2-297
Users, displaying the number of, 2-297
Users, information on, 2-283
Using COMI files, 2-45
Using COMO files, 2-48
Using SORT, 2-256
USRASR command, 2-297
UX_TAPE command, 2-297

V

V command, 2-301
Validation codes for login, 2-171
Variable files, global
see: Global variable files
Variables, global
see: Global variables
Verification options for wildcarding, 4-11
VERIFY function, 3-19
Verify mode, abbreviation, 2-8
-VERIFY wildcard option, 4-8
Verifying ARCHIVE or BACKUP tape catalogs, 2-115
version, 6-37
Virtual sense switches, 2-301
V-mode runfiles
see: EPFs; Runfiles
Volume name, definition of, 1-2
Volumes
see: Disks
VPSD command, 2-301
VPSD16 command, 2-301
VRPG command, 2-301
VRTSSW command, 2-301

W

-WALK_FROM treewalking option, 4-17
-WALK_TO treewalking option, 4-17
Warning prompt, changing, 2-214
Wild characters
on command-line, 1-8
table of, 4-5
WILD function, 3-20
Wildcard names, 4-5
Wildcard options, explanation of, 4-7
Wildcard options, use with LD command, 2-131
Wildcarding
date-selection options, 4-8
explanation of, 4-5
processing by PRIMOS, 5-6
type-designation options, 4-9
use with ABBREV command, 2-9
use with DELETE command, 2-75
use with LD command, 2-131, 2-137
use with LIST_MINI_COMMANDS command, 2-154
use with name generation, 4-23
use with SIZE command, 2-253
use with the WILD function, 3-20
use with treewalking, 4-12
verification options, 4-11
WORD command, 2-302
WP_ADMIN command, 2-302
WRITE_FILE function, 3-22
Writing magnetic tape labels, 2-129
Writing text into ASCII files, 3-22

X

XOFF/XON feature, enabling or disabling, 2-90, 2-288

Y

yank, 6-35, 6-37
yankreplace, 6-35, 6-37

Z

Zeroing memory, 2-105

READER RESPONSE FORM

PRIMOS Commands Reference Guide
DOC3108-7LA

Your feedback will help us continue to improve the quality, accuracy, and organization of our user publications.

1. How do you rate this document for overall usefulness?

   ☐ *excellent* ☐ *very good* ☐ *good* ☐ *fair* ☐ *poor*

2. What features of this manual did you find most useful?

3. What faults or errors in this manual gave you problems?

4. How does this manual compare to equivalent manuals produced by other computer companies?

   ☐ *Much better* ☐ *Slightly better* ☐ *About the same*
   ☐ *Much worse* ☐ *Slightly worse* ☐ *Can't judge*

5. Which other companies' manuals have you read?

Name:
Position:
Company:
Address:

Postal Code:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

First Class Permit #531 Natick, Massachusetts 01760

BUSINESS REPLY MAIL

Postage will be paid by:

Prime™

Attention: Technical Publications
Bldg 10
Prime Park, Natick, Ma. 01760