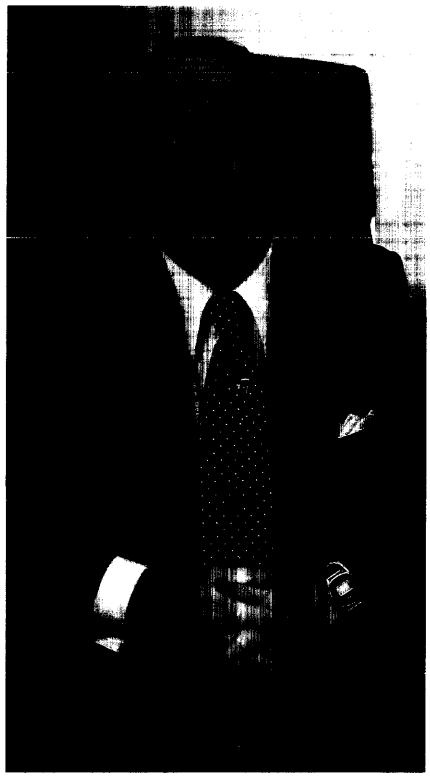
PRIMOS COMMANDS REFERENCE GUIDE



PRIME Computer

FDR 3108-101B

LS JUL 88

FDR-3108
JAN 81
REV. 18.

PRIMOS COMMANDS REFERENCE GUIDE

PRIME

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

First printing, January 1979
Second printing, revisions, May 1980
Third printing, revisions, January 1981

Production information: This book was composed in 10 and 11 point Melior by **J. L. Associates**. The covers were printed in 5 colors by **Mark-Burton**. Stock photography from **Four by Five**. The cover stock was 80# Cameo Dull Cover manufactured by **S. D. Warren Paper Company**. The text stock was 50# Mohawk Vellum Cream White manufactured by **Mohawk Paper Mills**. The text was printed in two colors by **Federated Lithographers**.

PRIMOS COMMANDS REFERENCE GUIDE

By Alice Landy

IMPORTANT NOTICE

This edition of the **PRIMOS Commands Reference Guide** represents a complete revision of the Rev. 17 (May, 1980) text. Changes in functionality and corrections made to the Rev. 17 text are indicated by a bar in the outer margin.

Additionally, several changes were made to the organization and content of the book. Section 1 contains the information previously in Sections 1 and 2 of the Rev. 17 text, Section 2 contains PRIMOS User Commands in alphabetical order, Section 3 is new with this edition and contains information on functions and variables, Section 4 remains the same, Appendix A is now RVEC parameters, Appendix B contains information on paper tape punch and copy utilities, Appendix C lists old commands, Appendix D is new and contains DMSTK formats, and Appendix E is new and contains both the ASCII and EBCDIC character sets.

At the back of this book you will find a copy of our Prime Technical Publications Survey. Please help us by completing and returning it to us.

1

INTRODUCTION

TO THE READER

This book is intended for the user who is working on a Prime computer and who needs detailed information on a particular PRIMOS command.

This is not an introduction to PRIMOS, Prime's operating system. Introductory material is supplied in **The Prime User's Guide.**

This book provides:

- detailed information on most PRIMOS commands that are 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 four sections.

1. Introduction
   - provides a brief review of the PRIMOS command line, and explains our conventions for displaying command line formats
   - provides a summary that lists and defines all PRIMOS commands, grouped by function
2. The 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 the detailed description may be found
3. Command Functions and Global Variables
   - explains the advanced command-line functionality provided by variables and function calls
   - lists and describes the functions available for use in the command line
4. FUTIL
   - provides a detailed description of Prime's file-handling utility

In addition, five appendices provide further details on miscellaneous matters such as debugging and paper-tape utilities.

PRIMOS COMMAND LINE FORMAT

The general format of the PRIMOS command line is:

COMMAND [names] [-OPTION argument] . . . [-OPTION argument]

For example, the format of the SPOOL command is:

SPOOL [pathname] [options]

In this example, the **pathname** tells PRIMOS the name of the file to be printed, while the **options** let the user specify how or where he or she wants the file printed. An example of a particular SPOOL command might be:

SPOOL MYFILE -FORM WHITE -AT BLDG2

COMMAND FORMAT CONVENTIONS

The conventions for PRIMOS command documentation are:

WORDS-IN-UPPERCASE: Capital letters identify command words or keywords. They are to be entered literally. If a portion of an uppercase word is in rust colored letters, the rust colored letters indicate the system-defined abbreviation.

Words-in-lowercase: Lower case letters identify arguments. The user substitutes an appropriate numerical or text value.

Braces { }: Braces indicate a choice of arguments and/or keywords. At least one choice must be selected.

Brackets [ ]: Brackets shown in brown indicate that the word or argument enclosed is optional. Brackets shown in rust indicate function calls. They must be entered literally.

Hyphen -: A hyphen identifies a command line option, as in: SPOOL -LIST. Hyphens must be included literally.

Parentheses ( ): When parentheses appear in a command format, they must be included literally.

Ellipsis ... : An ellipsis indicates that the preceding argument may be repeated.

Angle brackets <>: Angle brackets are used literally to separate the elements of a pathname. For example:

<FOREST>BEECH>BRANCH537>TWIG43>LEAF4

options: The word **options** indicates that one or more keywords and/or arguments can be given, and that a list of options for the particular command follows.

Spaces: Command words, arguments, and parameters are separated in command lines by one or more spaces. In order to contain a literal space, an argument must be enclosed in single quotes. For example, a pathname may contain a directory having a password:

'<FOREST>BEECH SECRET>BRANCH6'

The quotes ensure that the pathname is not interpreted as two items separated by a space.

CONVENTIONS IN EXAMPLES

In all examples, the user's input is rust-colored, and the system's output is not. For example:

```
OK, attach *>examples
OK, ed seginfo
EDIT
```

User input usually may be either in lowercase or in UPPERCASE. The rare exceptions will be specified in the commands where they occur.

MULTIPLE COMMANDS

Several commands may be given on a single line if the commands are separated by semicolons. For example:

```
ATTACH MYUFD; LISTF
```

If one command contains an error, PRIMOS still tries to execute the remaining commands. In the example above, if PRIMOS can't attach the user to MYUFD, it will still execute the LISTF, listing the contents of whatever directory the user is attached to.

An exception to this rule concerns the ABBREV command. Semicolons following ABBREV commands are interpreted literally, rather than as command separators. This allows abbreviations containing semicolons to be defined easily. For example, the command:

```
ABBREV -AC CA CLOSE ALL; RLS -ALL
```

defines one abbreviation (CA) that contains two commands.

For this reason, ABBREV commands cannot be followed on the same command line by another command. The second command will simply become part of the value of the abbreviation.

ADVANCED COMMAND LINE FUNCTIONALITY

In any PRIMOS command line, the commands, pathnames, options, or arguments may be supplied either literally or by means of:

- user-defined abbreviations
- user-defined global variables
- PRIMOS command functions.

User-defined abbreviations are explained in the discussion of the ABBREV command in Section 2. Global variables and command functions are discussed in Section 3.

When PRIMOS encounters a user-defined abbreviation, global variable, or function call on the command line, it substitutes the *value* of the abbreviation, variable, or function call for the abbreviation, variable, or function call itself. Values for abbreviations and global variables are retrieved from the user's abbreviation file or global variable file. Values for function calls are obtained by evaluating the function call at the time the command line is processed.

The Tilde

The user can force PRIMOS to ignore variables, command functions, and semicolons by beginning the command line with a tilde (~). When PRIMOS encounters a tilde as the first character in the command line, it removes the tilde and interprets the rest of the line literally: no variables, functions, or semicolons are processed. For example:

```
~ABBREV -AA DF %.FILE%.[DATE]
```

defines an abbreviation containing the literal string %.FILE%.[DATE]

In all other positions on the command line (including any position immediately following a semicolon), the tilde is interpreted literally.

SUMMARY OF PRIMOS COMMANDS

The following Summary of Commands lists all user commands and command functions commonly recognized by PRIMOS. In this summary, commands are grouped by function: file-handling commands, database commands, etc. All commands in this summary are discussed in more detail in Section 2. Command functions shown in this summary are discussed in more detail in Section 3.

A list of operator commands follows the summary of user commands. Operator commands are discussed in detail in **The System Administrator's Guide**.

Internal and External Commands

At the end of each definition in the Summary of Commands, you will find the word (Internal) or (External). This indicates whether the command is 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 UFD named CMDNC0. Some external commands invoke separately priced software products. Not all systems obtain all these products. 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.

If you need to know what external commands are available on your system, ATTACH to CMDNC0 and list its files with the LISTF command. (This works only if the System Administrator allows users to read CMDNC0.) An example of this procedure is:

```
OK, a <M180A1>cmdncØ
OK, listf

UFD=<M180A1>CMDNCØ  5  OWNER

MAGRST  VPSD    MAGSAV  FUTIL   SIZE    NUMBER  BASINP  PSD     FILVER
AVAIL   CRMPC   PRMPC   PRSER   SLIST   SPOOL   MDL     FILMEM  SEG
FIXRAT  EDB     MAKE    LOAD    SORT    PTCPY   COPY    PRVER   PROP
UPCASE  TRAMLC  ED      CX      VPSD16  HPSD    LABEL   PMA     PSD2Ø
TERM    CPMPC   MAGNET  CMPF    $$      MRGF    CONCAT  NSED    BATGEN
JOB     BATCH   PHYSAV  PHYRST  RUNOFF  LATE    COPY_DISK

OK,
```

SUMMARY OF USER COMMANDS

Gaining Access to and Exiting the System

LOGIN	Begins a terminal session. It identifies the user to the system and establishes the initial contact between system and user. (Internal)
LOGOUT	Ends a terminal session. It closes all files and releases the PRIMOS process to another user. (Internal)
ATTACH	Moves user's location from one directory to another. (Internal)

Directory Handling

CNAME	Changes the name of a directory. (Internal)
CREATE	Creates and names a new directory. (Internal)
DELETE	Deletes (removes) an empty directory from a UFD or sub-UFD. (Internal)
PASSWD	Replaces any existing passwords in the current directory with new passwords. (Internal)
LISTF	Lists the contents of the current directory. (Internal)
PROTEC	Defines rights of others to access the user's directory. (Internal)

File Handling

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)
CMPF	Compares two to five ASCII files; prints out any discrepancies it finds. (External)
CNAME	Changes the name of a file. (Internal)
DELETE	Deletes a file from a UFD or sub-UFD, freeing the storage space. (Internal)
FILVER	Compares run files, notifying the user of any differences between them. (External)
FUTIL	Invokes the FUTIL file utility that contains subsystem commands for copying, deleting, and listing files and directories. (External)
INPUT	Opens a source file for reading on file unit 1. (Internal)
LISTING	Opens a file for writing on file unit 2. (Internal)
MRGF	Merges two to five ASCII files, allowing user to resolve conflicts among them. (External)
NUMBER	Numbers or renumbers statements in a BASIC program. (External)
OPEN	Opens a file unit for reading, writing, or updating a specified file. (Internal)
PROTEC	Specifies access rights (read, write, delete) for owners and non-owners; may be applied to files or directories. (Internal)
SLIST	Prints the contents of a file on your terminal. (External)
SORT	Sorts up to 20 files into a single output file. (External)
SPOOL	Queues a file for printing or plotting on the system printers.
UPCASE	Creates an uppercase-only file from a file containing both upper and lowercase characters. (External)

Editors and Text Handlers

CONCAT	Combines a number of ASCII files into one; accepts formatting commands for spooling the file. (External)
ED	Invokes EDITOR, Prime's most commonly used text editor. (External)
EDB	Invokes Prime's binary editor (used primarily for building and maintaining subroutine libraries). (External)
NSED	Invokes the non-shared EDITOR (for use with PRIMOS II). (External)
RUNOFF	Invokes RUNOFF, Prime's text formatting utility. (External)

Compilers, Translators and Interpreters

BASIC	Invokes the Prime BASIC interpreter. (External)
BASICV	Invokes a virtual-memory BASIC subsystem (BASIC/VM). (External)
COBOL	Compiles a COBOL program. (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)
NCOBOL	Compiles a COBOL program, using the non-shared COBOL compiler. (External)
PASCAL	Compiles a Pascal program. (External)
PL1G	Compiles a PL/I, subset G, program. (External)
PMA	Assembles a program written in the Prime Macro Assembler language (PMA). (External)
RPG	Compiles an RPGII program. (External)
SPSS	Starts the Statistical Package for the Social Sciences (SPSS). (External)

Loading and Executing Programs

DELSEG	Frees previously used segments. (Internal)
FILMEM	Fills memory locations 100 to top of 32K with zeroes (used before invoking LOAD). (External)
LOAD	Invokes Prime's loader for R-mode object files. (External)
REN	Re-enters a subsystem following a QUIT or an error condition. (Internal)
RESTOR	Restores a runfile from disk to memory. (Internal)
RESUME	Executes a CPL program or an R-mode program. (Internal)
SAVE	Saves the contents of a specified location in memory. (Internal)
SEG	Loads and/or executes a V-mode or I-mode program. (External)
START	Starts a program loaded by the RESTOR command, or restarts a program halted by a QUIT or an error condition. (Internal)

Debuggers

DBG	Invokes the source level debugger, for debugging high-level language programs. (External)
PSD **HPSD** **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

COMINPUT	Executes a Command Input File. (Internal)
COMOUTPUT	Creates a record of command input and output. (Internal)
CPL	Executes a CPL program. (External)
JOB	Executes a CPL program or COMINPUT file as a Batch job. (External)
PHANTOM	Executes a CPL program or COMINPUT file as a phantom. (Internal)
$$	Used inside CPL or COMINPUT files to pass JOB information to the Batch monitor. (External)

Setting Your Terminal Characteristics

DELAY	Defines a time function that delays the printing of a character after a carriage return (CR) has been output to a terminal. (Internal)
LATE	Forces the terminal to ignore commands until a specified time. (External)
TERM	Defines terminal characteristics such as the erase character, break key, duplex and half-duplex operation. (External)

Defining Your Command Environment

ABBREV	Defines abbreviations for PRIMOS commands and their arguments. (Internal)
DEFINE_GVAR	Defines a global variable file. (Internal)
DELETE_VAR	Deletes one or more variables from a global variable file. (Internal)
LIST_VAR	Lists variables from a global variable file. (Internal)
RDY	Selects the prompt message you want displayed at your terminal. (Internal)
RLS	Frees space by discarding unwanted stack history. (Internal)
SET_VAR	Adds a variable to a global variable file. (Internal)

System Information

AVAIL	Provides information on amount of disk space being used. (External)
BATGEN	Provides information about Batch queues. (External)
DATE	Prints the current date and time at your terminal. (Internal)

PROP	Provides information about system printers and/or plotters. (External)
STATUS	Provides general information: what users are logged in, what disks are running, what nodes on a network are available, etc. (Internal)
USERS	Prints the number of users currently logged in. (Internal)

Information on Program and Command Execution

BATCH	Provides information on progress of user's Batch jobs. (External)
DMSTK	Produces a call/return trace of the user's stacks. (Internal)
PM	Prints the contents of the RVEC vector. (Internal)
PRERR	Prints locations and messages from PRIMOS's error vector, ERRVEC. (Internal)
TIME	Prints accounting information: time since login, CPU time used, and I/O time used. (Internal)

Message Facilities

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

Terminal I/O Handling

RSTERM	Empties the terminal's input and/or output buffers. (Internal)
TYPE	Prints text at the terminal or into a COMOUTPUT 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)
UNASSIGN	Releases control of a previously assigned peripheral device (Internal)

Magnetic Tapes:

LABEL	Creates either an ANSI COBOL level 1 volume label or an IBM 9-track EBCDIC or 7-track BCD label on a magnetic tape. (External)
MAGNET	Writes files or directories to tape or restores them from tape to disk. (External)
MAGRST	Restores files or directories from tapes created by MAGSAV. (External)
MAGSAV	Copies files or directories from disk to tape. (External)

Paper Tapes:

BASINP	Loads a BASIC source program from paper tape. (External)

MDL	Punches paper tape from specified sections of memory. (External)
PTCPY	Duplicates and verifies paper tapes, using the high-speed reader-punch. (External)

Punched Cards:

CPMPC	Punches a file onto a card deck. (External)
CRMPC	Reads 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 an assigned printer/plotter. (External)

Communications

NETLINK	Establishes a connection to any system on the Public Data Network. (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	Invokes PT45 interface program on systems where DPTX /DSC is running.
RJ1004 **RJ200UT** **RJ7020** **RJX80** **RJGRTS** **RJHASP**	Send utility commands for the Remote Job Entry system; used to submit jobs to remote computer sites. (External)
TCF	Invokes DPTX /TCF on a system where DPTX /TSF and DPTX /OSC are running. (External)
TRAMLC	Transmits or receives a file over an assigned AMLC line between two Prime computer systems. (External)
WS1004 **WS200UT** **WS7020** **WSX80** **WSGRTS** **WSHASP**	Control Remote Job Entry workstations. (External)

Data Management

DBMS Subsystems:

CDML	Invokes the COBOL data manipulation language. (External)
CLUP	Invokes the DBMS cleanup processor. (External)

CSUBS	Invokes the COBOL DBMS subschema. (External)
DBACP	Invokes the data base administrator command process. (External)
DBUTL	Invokes a data base dump utility that allows you to monitor the contents of a data base schema and shared user table. (External)
FDML	Invokes the FORTRAN DML preprocessor. (External)
FSUBS	Invokes the FORTRAN DBMS subschema. (External)
SCHDEC	Invokes the DBMS Schema Decompiler. (External)
SCHED	Invokes the schema editor (SCHED). (External)

MIDAS:

CREATK	Invokes a program to build a template for a MIDAS file. (External)
KBUILD	Invokes a program to build a keyed-index or direct access MIDAS file. (External)
KIDDEL	Invokes a program to delete all or part of the records in a MIDAS file. (External)
MCLUP	Invokes the MIDAS clean-up utility. (Exernal)
MPACK	Invokes a utility that packs and restructures MIDAS files. (External)

POWER:

POWER	Invokes the PRIME POWER data management facility. (External)

FORMS

FAP	Invokes the Forms Administrative Processor. (External)
FDL	Invokes the FORMS Definition Language. (External)

Command Functions

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 portion of a pathname.

EXISTS	Determines whether or not a file system object exists and whether its file type matches the type specified.
OPEN_FILE	Opens a file for reading or writing.
PATHNAME	Returns a full pathname.
READ_FILE	Reads a line from an ASCII file.
WILD	Produces a blank-separated list of entrynames representing the file system objects that match the specifications provided.
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 quotes around a string, and doubles any quotes appearing within the string.
SEARCH	Compares two strings; 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 quotes from around a specified text string and changes double quotes within string to single quotes.
VERIFY	Compares two strings; returns the position of the first character in one that does not match any character in the other.

Miscellaneous Functions:

CND_INFO	Allows a CPL condition handler to examine the condition information of the most recent condition on the stack.
DATE	Returns 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 specified text on the terminal screen and waits for a YES or NO reply.
RESCAN	Removes one level of quotes from a specified string and evaluates any function calls or variable references that no longer appear in quotes.
RESPONSE	Prints a specified prompt at the terminal and waits for any reply.

System Settings

ASRCWD	Directs output stream to terminal or peripheral device. (Internal)
SVCSW	Controls handling of SVC instructions. (Internal)
VRTSSW	Sets the virtual sense switches. (Internal)

SUMMARY OF OPERATOR COMMANDS

Below are the PRIMOS commands used primarily by the System Operator /Administrator. Many of these commands can be issued only from the Supervisor terminal. These commands are explained briefly in Section 2; for a full explanation, see **The System Administrator's Guide.**

ADDISK
AMLC
BATCH
BATGEN
CHAP
CONFIG
COPY_DISK
DISKS
DPTCFG
DPTX
ELIGTS
FIXRAT
LOGPRT
LOOK
MAKE
MAXSCH
MAXUSR
MCLUP
NET
NETCFG
OPRPRI
PHYRST
PHYSAV
PROP
REMOTE
REPLY
SETIME
SETMOD
SHARE
SHUTDN
STARTUP
USRASR

COMMANDS

▶ ABBREV [pathname] [options]

The ABBREV command allows users to create their own abbreviations for PRIMOS commands and their arguments and to use these abbreviations at will during interactive sessions and in CPL programs. Abbreviations *cannot* be defined for subcommands. They may be used in CPL programs, if CPL's &EXPAND directive is used; but they may not be used in command input files. (System administrators may disable the abbreviation preprocessor, thus preventing the use of ABBREV files at their installations.)

When an abbreviation file is active, PRIMOS calls its abbreviation preprocessor to scan each command line entered from the terminal. The abbreviation processor checks each token (that is, each word or numeral) and expands it to its full form before passing the expanded command to the standard command processor for execution. Abbreviations are not expanded when commands are read from a command input file.

Using ABBREV

To use ABBREV:

1. Create an empty abbreviation file with the command **ABBREV new-pathname -CREATE**. (If the UFD in which the file is created has a password, the password must be part of **new-pathname**.) For example:

```
abbrev 'beech secret'>abbrevs -create
```

2. Define abbreviations within the file up to a limit of approximately 200 abbreviations.
3. At the start of a subsequent session, reactivate the abbreviation file by giving the command **ABBREV pathname.** If you deactivate the file during the session (via the **ABBREV -OFF** command), reactivate it with the command **ABBREV -ON**.
4. If you are in a multi-user environment, deactivate your abbreviation file at the end of your work session, either by giving the **ABBREV -OFF** command or by logging out.

Note

Certain error conditions, which return the message "User Environment Re-Initialized," wipe out your active abbreviations. Restore them by giving the command **ABBREV pathname**.

Rules for defining abbreviations

Each abbreviation has two parts: its **name** and its **value**. Rules for each are as follows:

Names Names may be up to eight characters in length. They may contain any ASCII character except spaces, tab characters, quotes ('), commas, greater-than symbols (>), and vertical bars (|). They may be given as lowercase or uppercase characters; the abbreviation preprocessor converts all lowercase characters to uppercase.

Values The value of an abbreviation is the character string represented by the abbreviation. All ASCII characters are legal within values, including spaces. Leading spaces and/or tabs are removed during definition. The percent sign (%) functions as an escape character and is used to define variables. The token %i% in a value stands for the *ith* token found after the abbreviation in the command line. Currently, **i** can range from 1 to 9. The use of variables permits the repetition and reordering of tokens from the command within the expanded value. For example, defining the abbreviation:

```
cmpl %1% %2%.%1% -L %2%.LIST -B %2%.BIN -64V -DEBUG -EXPLIST
```

would allow the command:

```
cmpl pl1g foo
```

to be expanded to:

```
PL1G FOO.PL1G -L FOO.LIST -B FOO.BIN -64V -DEBUG -EXPLIST
```

If a command line contains fewer variables than the value calls for, the missing variables will be taken to be null strings. If a command line provides more variables than the value expects, the unexpected variables are added to the end of the command line. Thus, if the value of X is %1% %4%, then the command X A B C D E is expanded to A D E.

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.

Here is an example of creating and using two abbreviations. (We will work in VERIFY mode, explained below, to make ABBREV echo the expanded abbreviations before using them.)

```
OK, abbrev abbrevs -create
Creating new abbreviation file: <FOREST>BEECH>ABBREVS (ab_file_)
OK, abbrev -add u <forest>beech>branch5>%1%
OK, abbrev -add sp spool %1% -at 1 %2% %3% -list
OK, abbrev -list
Abbreviation file: <FOREST>BEECH>ABBREVS
Abbreviations: 2

    SP       spool %1% -at 1 %2% %3% -list
    U        <forest>beech>branch5>%1%

OK, ab -verify
OK, sp u squirrel -defer 2200
(listen_) "spool <forest>beech>branch5>squirrel -at 1 -defer 2200 -list"
[SPOOL rev 18.0]
PRT001 spooled,  records:    1, name: SQUIRREL

 user  prt  time  name              size  opts/#   form  defer  at: CAROUSEL
------ --- -----  ---------------- -----  ------  ------ -----  --------------
BEECH  001 13:42  SQUIRREL             1                22:00  1
OK,
```

Since U is the inner abbreviation in this example, it is expanded first. It wants one variable, %1%; so it takes S1 as that variable. When SP is expanded, it takes the expanded value of U (<FOREST>BEECH>BRANCH5>SQUIRREL) as its %1%, then takes -DEFER and 2200 as its %2% and %3%.

Abbreviations may be defined for ABBREV commands. The abbreviation 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. For example, **.A** is a handy abbreviation for ABBREV -ADD COMMAND, and **.L** is a handy abbreviation for ABBREV -LIST.

```
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>ABBREVS
Abbreviations: 4

(C) .A      abbrev -ac
(C) .L      abbrev -list
    SP      spool %1% -at 1 %2% %3% -list
    U       <forest>beech>branch5>%1%

OK,
```

Using abbreviations in command lines

- 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 quotes and typed in uppercase.
- Tokens in a command line (i.e., abbreviations, commands, or arguments) may be separated by spaces, tabs, commas, or >.
- Any 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.
- Expansion of any command line may be suppressed by the -EXECUTE option, explained below.

Options

The ABBREV command supports the options listed below. In each case, supplying the pathname within the command activates the file as well as performing the option-related task. If the file is already active, the pathname is not needed.

To create a file:

-CREATE — Creates and activates an empty abbreviation file. A pathname must be supplied with this option. If file named by the pathname already exists, the command activates that file.

To activate and deactivate files:

ABBREV pathname	Activates an existing abbreviation file. Naming a non-existent file, or a file that is not an abbreviation file, produces an error message.
-OFF	Turns off abbreviation expansion.
-ON	Turns on abbreviation expansion. If no pathname is given, the command reactivates the file the user was previously using. If pathname is given, activates that file.
-VERIFY	Turns on verify mode. This prints each expanded command line at the terminal before executing it.
-NO_VERIFY	Turns off verify mode (default).
-EXECUTE rest-of-line	Passes **rest-of-line** to command processor for execution without expanding it.
-EXPAND rest-of-line	Expands **rest-of-line** and prints it out on terminal, but does not execute expanded line.

To monitor file:

-LIST [name-1 ... [name-n]]	Lists the specified abbreviations from the current abbreviation file. If no **names** are given, this option lists the complete file.
-STATUS	Prints out the name of the current abbreviation file and the number of abbreviations it contains.

To add, change, or delete abbreviations:

-ADD name value	Adds the abbreviation **name** to the current file and gives it the value **value.** If an abbreviation for **value** already exists, the user will be asked if he wants to replace it.
-ADD_ARGUMENT name value	Adds an abbreviation that will be expanded only when it occurs in the argument position of a command line.
-ADD_COMMAND name value	Adds an abbreviation that will be expanded only when it occurs in the command position of a comand line.
-CHANGE name-1 [...name-n]	Changes the specified abbreviations so that they are expandable anywhere on the command line.
-CHANGE_ARGUMENT name-1 [...name-n]	Changes the specified abbreviations so that they are expandable only in the argument position on the command line.
-CHANGE_COMMAND name-1 [...name-n]	Changes the specified abbreviations so that they are expandable 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.**
-DELETE name-1 [...name-n]	Deletes the specified abbreviations from the abbreviation file.
-NO_QUERY	Replaces old abbreviation without asking. (Only useful if followed by one of the ADD commands.)

A **wildcard** may be used with the ABBREV command. Wildcards provide an easy way of selecting a set of abbreviation names. When you use a wildcard in place of a name, ABBREV selects all abbreviation names that match the wildcard.

A wildcard is formed from one or more wildcard characters (which "match" any other characters, according to the rules specified below), plus one or more regular characters (which match only themselves). Thus, the wildcard name **a@a** uses the regular character **a** and the wildcard character @. It will match any abbreviation that begins and ends with **a** no matter what characters are in the middle.

The following rules apply to wildcards:

- @ matches zero or more characters in the corresponding name or component
- @@ matches zero or more characters across components
- + matches any single character in the corresponding components (i.e., does not match component separator characters)
- Λ The inverted match character selects the subset of objects whose names *do not* match the rest of the wildcard name. Only one inverted match character per wildcard name is permitted.

For example, if you had an abbreviation file containing the following abbreviations:

```
(C) .A      abbrev -ac
(C) .D      ab -delete
(C) .L      abbrev -list
    SP      spool %1% -at 1 %2% %3% -list
    U       <forest>beech>branch5>%1%
```

You could list all names beginning with the character "." by giving the command ABBREV -LIST .@ thus:

```
OK, abbrev -list .@
(C) .A      abbrev -ac
(C) .D      ab -delete
(C) .L      abbrev -list
OK,
```

▶ ADDISK [PROTECT] pdisk-1 [pdisk-2...pdisk-n] *Operator command*

ADDISK starts up the disk specified by the physical disk **pdisk** parameters. For details see **The System Administrator's Guide.**

▶ AMLC [protocol] line [config] [lword] *Operator command*

The AMLC command starts up an AMLC line. The parameters are discussed in detail in **The System Administrator's Guide.**

▶ ASRCWD [number]

The chief use of ASRCWD is to allow a user with a serial line printer to recover from the following situation: Assume output is being sent to the serial line printer and a user program aborts or a CONTROL-P condition occurs. At this point, the user is not able to get output (from the Editor, for example) printed or displayed at the user's terminal. Issuing the command line: ASR 0 then allows the user to recover and get output at the terminal.

For further information, see ASRCWD in **Old Commands, Appendix C.**

▶ **ASSIGN** { **device [-WAIT]** | **DISK pdisk [-WAIT]** | **AMLC [protocol] amlc-line [config]** }

The ASSIGN command obtains complete control over a disk or a peripheral device from the user terminal.

device is an available device. These are the assignable devices:

Device code	Meaning
CARDR	Serial Card Reader
CRn (0≤n≤1)	MPC Parallel Card Reader or Reader/Punch
DISK pdisk	Physical Disk Partition (**pdisk** is a partition (volume) number)
GS0 - GS3	Vector General graphics display terminal
MG0 - MG3	Megatek graphics display terminal
MTn (0≤n≤7)	Magnetic Tape Unit
PRn (0≤n≤3)	Line Printer
PTR	Paper Tape Reader
PUNCH	Paper Tape Punch
PLOT	Printer/Plotter
SMLCnn	Synchronous Communications Line (00≤nn≤07)

If the device is currently assigned to another user, the system replies:

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

unless the optional argument **-WAIT** was supplied. In this case, the ASSIGN command is queued until the device is UNASSIGNed by another user, or until the other user presses the CONTROL-P or BREAK key, or logs out. The terminal from which the ASSIGN command is invoked is unavailable for use until the device assigned is again available for assignment, or until the CONTROL-P or BREAK key is pressed by the user at that terminal.

Examples:

```
ASSIGN CENPR -WAIT
```

assigns the serial line printer and queues the assignment if the printer is already assigned.

```
AS PTR
```

assigns the paper-tape reader.

If the user does not ASSIGN a device and attempts to perform I/O to or from the device, the error message:

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

is printed at the terminal.

ASSIGNING MAGNETIC TAPES

For magnetic tapes, the ASSIGN command either indicates by number which physical tape drive the user wants, or provides a description of a tape drive that will meet the user's requirements. Additionally, ASSIGN allows special requests to be made of the system operator; for example, removing the WRITE-ring or mounting a particular tape. (These requests are useful primarily for Batch jobs.) The command format, complete with optional arguments, is:

ASSIGN { **MTpdn [-ALIAS MTldn]** / **MTX -ALIAS MTldn** } **[options]**

options are:

-density
-protect
-TPID id
-track
-WAIT

The options and arguments are described below:

Option	Description
MTpdn	Magnetic tape (MT) unit number from 0 to 7, inclusive. **pdn** is the physical device number assigned to each drive at system startup.
MTldn	The logical drive number, from 0 to 7, inclusive. **ldn** is a user-specified number assigned to particular physical drive unit; mapped into **pdn** in subsequent magnetic tape operations.
MTX	Indicates "any available drive"; MUST be accompanied by -ALIAS MTldn option. The particular drive assigned depends on any other options that appear on the command line.
WAIT	Indicates user is willing to wait until requested drive is available.
TPID	Indicates that a tape "id" is to follow. Used to request mounting of a particular tape; requires operator intervention.
id	A list of tape identifiers (arguments) describing a particular reel of tape, and/or type of tape drive (name, number, etc.). Identifiers must not contain the following delimiters: commas, spaces, .NL. and/*. They may not begin with a hyphen or dash (-), which is a reserved character indicating the next control argument on the ASSIGN statement line.
protect	Protection rights may be either: **RINGON** (read and write) *or* **RINGOFF** (read only, write protect) Requires operator intervention.
density	Particular tape density settings, including: **556 [BPI]** **800 [BPI]** **1600 [BPI]** **6250 [BPI]** Most drives can handle 800 and 1600 bpi settings. Requires operator intervention.

Examples:

```
OK, AS MT1
Device MT1 assigned.
OK,
```

Magnetic tape drive MT1 is assigned (1 is the physical device number).

```
OK, AS MT1 -ALIAS MTØ
Device MT1 assigned.
OK,
```

Physical device MT1 will now be referred to as logical MT0. The physical-to-logical number correspondence can be listed with the STAT DEV command:

```
OK, AS MT1 -ALIAS MTØ
Device MT1 Assigned.
OK, STAT DEV

DEVICE  USRNAM  USRNUM  LDEVICE
MT1     BEECH    49     MTØ


OK,
```

```
assign mtx -alias mt4
```

The operator is requested to assign any available tape drive as logical device 4. A message is displayed at the user's terminal, indicating which physical drive has been assigned. For example:

```
Device MTØ Assigned.
```

The operator has assigned magnetic tape drive MT0 in response to the above request.

```
as mtx -alias mt3 -tpid power -9trk -ringoff -625Ø
```

The operator is requested to mount the "POWER" tape (a 9-track tape) on any drive that can handle 6250 bpi. In addition, the user wants write-protection and is assigning an alias of MT3 (ldn) to whatever device the operator chooses.

This request, if processed, might be acknowledged with this display:

```
Device MTØ Assigned.
```

If a request cannot be handled by the operator for any reason, the following message appears at the terminal:

```
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; 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 will result in the message:

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

DISKS: In assigning disks, **pdisk** is a physical disk number (as printed by STATUS DISKS). A user may only ASSIGN a disk that is not already assigned and that appears in 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 supervisor terminal operator's knowledge, or from assigning disks or partitions the operator wishes to reserve for special use. For a disk to be ASSIGNed to a user, it must not be the paging disk, nor ASSIGNed to another user, nor a disk specified in a previous STARTUP command. To ASSIGN a disk that has been started by STARTUP or ADDISK, it must first be shut down at the supervisor terminal by the SHUTDN command.

For complete disk-assignment details, refer to **The System Administrator's Guide.**

```
AS DISK 46Ø

AS DISK 54
```

Each of the above command lines assigns a disk partition. The number is the partition (volume) number.

The maximum number of disk partitions that may be ASSIGNed to all users at any one time is ten. If an attempt is made to ASSIGN too many disks, the message:

```
ASSIGN TABLE FULL
```

is printed.

AMLC lines: A user may assign an AMLC line as follows:

ASSIGN AMLC [protocol] line [config]

where **line** specifies a line number. Using this form of the ASSIGN command, a user may assign the AMLC line number and may set a terminal protocol and line configuration word for the line specified by the line number. Refer to the description of the AMLC command in **The System Administrator's Guide** for a complete description of possible parameters for the arguments **protocol** and **config.** Values for protocol and config are summarized below.

Protocol	Meaning
TTY	Normal terminal
TTYHS	TTY with per-character interrupt
TRAN	Transparent (no character conversion)
TRANHS	TRAN with per-character interrupt
TTYUPC	TTY with lower-to-uppercase translation for output
TTYHUP	High-speed TTYUPC

Config	Meaning
2033	110 Baud
2213	300 Baud
2313	1200 Baud (Default)
2413	Programmable clock (default = 9600)

A user may only ASSIGN an AMLC line if it has been configured to be ASSIGNed, and if it is not ASSIGNed to another user.

A user terminal line may be ASSIGNed if, first, the command:

AMLC TTYNOP amlc-line

has been given at the supervisor terminal.

SMLC lines: A user may assign an SMLC line as follows:

ASSIGN SMLCnn [-WAIT]

where **nn** is the SMLC line number between 00 and 07.

▶ ATTACH name [passwd]

The ATTACH command changes the user's working directory to **name**.

Note

An obsolete form of the ATTACH command used **ldisk** and **key** parameters to keep the user's "home" and "current" directories separate. This version of the ATTACH command is documented in Appendix D, OLD COMMANDS.

Changing the working directory: After logging in, the user's working directory is set to the login UFD by PRIMOS. The user can move (i.e., ATTACH) to another directory in the PRIMOS tree structure with the ATTACH command. The format is:

ATTACH new-directory

new-directory is the pathname of the new working directory.

If new-directory has a password, type the password after the directory name. For example:

```
ATTACH BEECH SECRET
```

Passwords may be specified in lower and uppercase. To specify a lowercase password enclose the password in single quotes. Otherwise, the lowercase characters of the password will be converted to uppercase by the system. For example:

```
OK, ATTACH BEECH 'secret'
```

To specify an uppercase password type in the password in either upper or lowercase but do not enclose the password in single quotes. Lowercase characters not enclosed in quotes will be converted to uppercase by the system.

If a password is included in a pathname as in:

```
OK, ATTACH 'BEECH SECRET>BRANCH5'
```

the entire pathname is included in single quotes. In this instance lowercase passwords will not be converted to upper case. You must enter the password in either upper or lowercase characters as desired.

To set the MFD of a disk as the working directory, the format is slightly different.

ATTACH '<volume>MFD mfd-password'

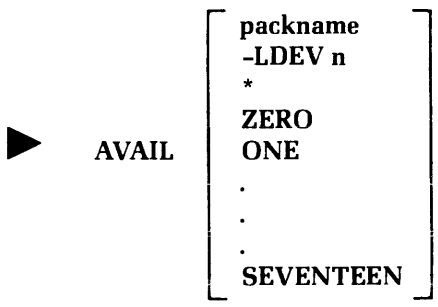
volume is either the literal volume name or the logical disk number, and **mfd-password** is the password of the MFD. A password is always required for an MFD.

Recovering from errors while attaching: If an error message is returned following an ATTACH command (for example, if a UFD is not found), the user remains attached to the previous working directory.

Several examples show all ordinary ATTACHments:

```
OK, attach <cooper>sport
OK, cr baseball
OK, attach sport>baseball
OK, create gehrig
OK, create ruth
OK, listf

UFD=<COOPER>SPORT>BASEBALL     5  OWNER

GEHRIG  RUTH
```

```
OK, attach *>ruth
OK, create pitching
OK, create hitting
OK, listf

UFD=<COOPER>SPORT>BASEBALL>RUTH    5  OWNER

PITCHING         HITTING

OK, attach '<1>MFD XXXXXX'
OK, listf

UFD=<COOPER>MFD     1  NONOWN

COOPER MFD  BOOT  PATS  VISITS. SPORT
OK,
```



The AVAIL command prints disk usage statistics at the terminal. These statistics give the number of disk records available for use in the specified logical disk. Numbers are given in decimal; records are "normalized" (880 byte) records.

If no argument is specified, AVAIL types the number of available records on the current logical disk and the percentage of space used up on that disk, provided that the nonowner password on the MFD is XXXXXX and that nonowners are allowed read access. If these conditions are not met, the message INSUFFICIENT ACCESS RIGHTS is returned when AVAIL is invoked with no argument. Thus, if the current disk was logical disk three, with the **packname**DOCUMN, AVAIL would produce the following message:

```
OK, avail
     VOLUME DOCUMN
      34476 TOTAL RECORDS (NORMALIZED)

      13565 RECORDS AVAILABLE (NORMALIZED)
      60.7

OK,
```

Users can also check availability by specifying either the packname or the logical disk number of a disk. (Use of the packname depends on the MFD nonowner password being set to XXXXXX; use of the logical disk number does not.) The number may be given either as a decimal number with the **-LDEV** option (e.g., -LDEV 3), or by spelling out the number (e.g., THREE). For example:

```
OK, avail documn
     VOLUME DOCUMN
      34476 TOTAL RECORDS (NORMALIZED)
      13579 RECORDS AVAILABLE (NORMALIZED)
      60.6

OK,


OK, avail three
     VOLUME DOCUMN
      34476 TOTAL RECORDS (NORMALIZED)
      13574 RECORDS AVAILABLE (NORMALIZED)
      60.6

OK,
```

Users can check availability and status of all started disks by giving AVAIL with an asterisk (*) as argument:

```
OK, avail *

VOLUME      TOTAL       FREE    %
  ID        RECS        RECS   FULL     COMMENTS
-------------------------------------------------
FOREST       34476       3202   90.7      460
OCEAN        68952       7270   89.5    11060
DOCUMN       34476      13570   60.6    30460
ATLANT       68952      25774   62.6    41060
PLAIN       120666       5469   95.5    61461
MOUNTS      172381       1570   99.1    12462
REEFS        34476      28730   16.7    60462
DESERT      327524       1915   99.4     4463
CITIES       34476      14519   57.9      460

OK,
```

If the system administrator has not set up the file SYSTEM>DISCS, the AVAIL * command will print the error message "NO DISCS FILE IN UFD SYSTEM".

If the MFD owner password is not XXXXXX, the AVAIL command will not work under PRIMOS II. Under PRIMOS, AVAIL will work with the **packname** and * optional arguments if: the MFD owner or nonowner password is set to XXXXXX and the **packname** (name of DSKRAT file) in the MFD has nonowner PROTECtion rights of 1.

See also: **STATUS.**

▶ BASIC [pathname]

The BASIC command loads the Prime BASIC language interpreter. Single-precision arithmetic is standard. For further information, refer to **The Interpretive BASIC User Guide.**

When BASIC is invoked with the **pathname** parameter, BASIC loads and runs the contents of pathname as a BASIC language program, then returns to PRIMOS command level. BASIC

invoked without a **pathname** or **filename** starts the BASIC interpreter, for editing or other user-specified BASIC operations.

See also: **DBASIC, BASICV, NUMBER.**

▶ BASICV [pathname]

BASICV invokes a virtual-memory BASIC subsystem (BASIC /VM). If a **pathname** is given, BASICV takes control and runs the contents of pathname as a BASIC-language program. If no pathname is given, BASIC /VM assumes control and prints a prompt character and waits for input. For example:

```
OK, BASICV
BASICV REV18.1
>NEW MUMBLE
>QUIT
OK,
```

For further information refer to **The BASIC/VM Programmer's Guide.**

▶ BASINP pathname

The BASINP command invokes a program that loads, from paper tape, a BASIC source program written for a computer system other than a Prime computer. **pathname** is the name of the file into which the contents of the paper tape are to be read.

See also: **BASIC**

▶ BATCH -DISPLAY

Users can call the BATCH -DISPLAY command to see information on jobs being processed by the Batch subsystem. Information displayed includes the number of jobs waiting in each Batch queue, and the user-name, job-id, user number, and queue for each currently executing job.

See also: **JOB, BATGEN**

▶ BATCH SYSTEM { -START / -STOP } *Operator command*

The BATCH operator's command tells the Batch monitor to start (or stop) processing users' jobs.

▶ BATGEN { -STATUS / -DISPLAY }

BATGEN -STATUS lists name and status (i.e., blocked or unblocked) for each Batch queue. An unblocked queue accepts jobs; a blocked queue does not.

BATGEN -DISPLAY displays queuename, status, and characteristics for each queue.

BATGEN accepts other arguments, but these are used only by the System Administrator to define, modify, or delete queues. For details, see **The System Administrator's Guide.**

Note

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

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 is assigned the name **pathname**. If pathname is a simple file name, it will be opened in the current directory. This command has the same effect as OPEN **pathname 3 2**.

The F77, PL1G, FTN, COBOL and RPGII compilers, and the PMA assembler, normally open a file named filename . BIN or B_filename as the binary output file (where filename is the source filename). A BINARY command is useful if the user wants to send the output of several compilations to a single file.

▶ CDML

CDML invokes the COBOL data manipulation language. CDML is an external command. For further information, refer to **The DBMS COBOL Subschema Guide.**

▶ CHAP { -nn / [ALL] } [priority] [timeslice]

Operator command

CHAP is an internal command that changes a specified user's hardware priority and the amount of time that user has for processing (timeslice). A user's default priority is 1, in a range of 0 (lowest) to 3 (highest). CHAP is an operator command.

Note

The timeslice or priority of user number 1 cannot be modified.

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

The CLOSE command closes the named file **pathname**, specified file units, or **ALL** file units. Each **funit** is an octal number with a current range of '1 to '177. The form: **CLOSE ALL** closes all files and units from to '1 to '176; it does not close file unit '177. (In a command file or CPL program, specify each item to be closed; do not use CLOSE ALL or the command file itself will be closed.)

The CLOSE ALL command also makes sure that buffers are retrieved properly and resets the state of the file system. CLOSE ALL is sometimes useful when the user is uncertain as to the state of the files in the current directory.

If a program is stopped with a BREAK, then the CLOSE ALL command should be given. If CLOSE ALL is not given, a difficulty such as

```
File in use. OMEGA (OPENR)
File open on delete. OMEGA
ER!
```

may arise with the next command.

Once CLOSE ALL has been given, the stopped program cannot be continued (STARTed).

If pathname cannot be found, an error message is printed and the CLOSE command returns to PRIMOS command level. Closing an already-closed file unit is not an error.

When issued from the supervisor terminal, the CLOSE command closes access to the specified files for all users.

See also: **OPEN**

▶ CLUP

CLUP is the DBMS Cleanup Processor command. See **The DBMS Administrator's Guide.**

▶ CMPF file-a file-b [file-c...file-e] { [-BRIEF] [-MINL number] [-REPORT pathname] }

The CMPF command allows the user to compare up to five files containing ASCII text lines of any length. The files may be specified by pathnames. One file, **file-a**, is treated as the original file. The CMPF command produces output that shows text lines that were (1) added; (2) changed from a previous line or lines; or (3) deleted from the original file, and not present in the new files.

The CMPF command, along with the MRGF command, helps ease the problems of parallel software development.

See also: **FILVER, MRGF**.

file-a	The treename of the original file (i.e., the file that is to be treated as the common ancestor of files file-b through file-e).
file-b, **file-c,** **file-d,** **file-e**	Optional additional files to be compared with **file-a.**
-BRIEF	Suppresses the printing of differing lines of text within the special files at the terminal. Instead, only the file identification letters and line numbers are printed.
-MINL number	Sets the minimum **number** of lines that must match, after a difference in the files being compared, in order to resynchronize all file comparison. The default value is: -MINL 3.
-REPORT pathname	Produces a file, **pathname**, containing the differences, instead of printing (or displaying) them at the terminal.

CMPF operation: When a difference is found between file-a and any other file specified, CMPF attempts to get the files being compared back into synchronization. The process is accomplished only when a certain minimum number of lines (default=3) match in all files that were specified in the CMPF command line. After resynchronization is complete, lines that differ between file-a and any of the other specified files are printed at the terminal (or written into the report file if the -REPORT option was specified). Then, the comparison between files continues.

When the differences between files are printed (or displayed, or written), each line from file-a is identified by the letter A and the line number of that line. Lines of file-b, file-c, file-d, and file-e are similarly printed, using the letters B through E respectively.

Consider the following two files:

```
FILEA       FILEB

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

A CMPF comparison of these two files works as follows:

```
OK, CMPF FILEA FILEB

A2        quick
A3        brown
CHANGED TO
B2        swift
B3        red


A8        lazy
DELETED BEFORE
B8        dog

COMPARISON FINISHED.
2 DISCREPANCIES FOUND.

OK,
```

▶ CNAME oldname newname

CNAME changes the name of a file (or directory) from **oldname** to **newname**. Oldname may be specified by pathname; newname may not. When a pathname has been used for oldname, newname replaces only the final element of the pathname. The user is cautioned not to change names of special UFDs such as CMDNC0. CNAME requires owner status to the directory.

Examples:

```
OK, CNAME OSS3 OLD.S3
OK,
```

changes the name of a file from OSS3 to OLD.S3.

```
CNAME <USER>SPARE2 JHNDOE
```

changes the name of the UFD SPARE2 in volume USER to JHNDOE.

▶ COBOL pathname [options]

The COBOL command loads the Prime COBOL compiler and compiles the object program from an ASCII source file, pathname.

For complete details, default values and examples, see **The COBOL Reference Guide.**

See also: **NCOBOL.**

▶ COMINPUT [pathname] [funit] { -CONTINUE | -END | -PAUSE | -START | -TTY }

The COMINPUT command causes PRIMOS to read its input stream from a specified file rather than from the user's terminal. Commands are processed as if they were entered at the terminal.

pathname	The file from which input is to be read. If it is only a filename, not a **pathname**, it will be found in the current directory.
funit	The PRIMOS file unit number on which the input file is to be opened. If omitted, File Unit 6 is assumed.
-TTY, -END, -PAUSE, -CONTINUE, and -START	Specify control flow. Their effect is discussed in the following paragraphs.

Note

The COMINPUT command must be specified with at least one parameter. If CO is specified with a null parameter, the message NOT FOUND is printed at the terminal.

Command input files are especially useful for repetitive processes such as compiling and loading a series of programs, building libraries, running production jobs, etc. Command files are normally constructed with the text editor, ED.

PRIMOS reads commands from the command input file, **pathname**, by opening the specified file unit and reading, then processing the series of commands, one line at a time.

When the command line:

CO -TTY or CO -END

is encountered, PRIMOS switches the input stream and takes subsequent commands at the terminal. If the user has specified a file unit (**funit**) other than the default Unit 6 when COMINPUT was invoked, then it is necessary to specify a **funit** using the form:

COMINPUT -TTY funit

The form:

COMINPUT -PAUSE

leaves the current command input unit (**funit**) open and returns to command level. Thus, a user can invoke other commands or use **COMINPUT pathname funit** to start another command file on another unit, before issuing a COMINPUT -CONTINUE [funit] line to continue the original command file.

The command line:

COMINPUT -START [funit]

may be used to restart processing of a command file following either a BREAK (or CONTROL-P) or a warm start of the PRIMOS system.

The -START option must not be used if PRIMOS command level has been entered from a command file process by means other than a QUIT; for example, by an error message and ER!. In these instances, to continue processing of a command file, the command:

COMINPUT -CONTINUE [funit]

should be issued.

It is advisable to use COMINPUT -START when the quit is from inside a processing program, and to use 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 the user to reinvoke the command that caused the error message and, subsequently, resume the command file at the command following the one that caused the error.

CAUTION

The use of the command **CLOSE ALL** within the command file closes the command input file unit (funit) and causes the message:

```
Unit not open.  Cominput (Input from terminal)
```

to be displayed (or printed) at the terminal.

Comments: Comments may be inserted in a command input file. The comment format is:

/*string

The characters /* begin the comment, which ends at the end of the line. A comment may also be appended to a command line. For example:

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

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

Chaining command files: The -CONTINUE option allows command files to be chained. The last command file in the chain must be terminated with a CO -TTY (or CO -END) to ensure that files opened in the process are closed and that the chain is terminated properly. Furthermore, the user must be careful to keep track of what file units are being opened and closed as the chain progresses, especially those file units that are opened for command input files.

See also: **CPL**

▶ COMOUTPUT [pathname] [options]

The COMOUTPUT command causes PRIMOS to direct its output stream to a specified command output file, **pathname**, as well as the user's terminal. If the file is specified by a **filename**, rather than an entire **pathname**, it will be placed in the current directory.

COMOUTPUT is not available under PRIMOS II.

The options are described below. Logical combinations of options are permissible (refer to the examples in this section).

-CONTINUE	Continues command output to a file. With the -CONTINUE option, subsequent terminal output is appended to the file specified by **pathname.**
-END	Stops command output to a file and closes the command output file unit.
-NTTY	Turns off the terminal output (i.e., does not print or display responses to command lines, including the prompt OK). Once -NTTY has been specified, terminal output is not turned on until either -TTY is specified in a subsequent COMOUTPUT command, an error occurs, or the break key is hit.
-PAUSE	Stops command output to **pathname.** However, the command output file, **pathname**, remains open.
-TTY °	Turns on the terminal output.

Command output files are useful when the user desires to keep a record of his terminal transactions in the sytem. PRIMOS writes all command input and output responses on the file specified by **pathname**. The file is opened on File Unit 177 (or the highest file unit provided by the System Administrator).

Examples:

```
OK, COMO OUTPUT
```

This command line arranges for subsequent terminal output to be written into the file named OUTPUT. Commands and echoed responses continue to be displayed at the terminal. The file named OUTPUT is overwritten if it already exists.

```
OK, COMO -NTTY
```

Terminal output continues to the file named OUTPUT, but no terminal output is printed or displayed at the terminal.

```
OK, COMO -END
```

The file named OUTPUT is closed. Furthermore, since -TTY was not specified following a -NTTY, terminal output will not be printed or displayed until the command line:

```
COMO -TTY
```

is issued.

```
OK, COMO OUTPUT -C -P
```

The file named OUTPUT is opened and positioned to end-of-file, but no terminal output is sent to the file.

▶ CONCAT [pathname] [options]

CONCAT is used to combine a number of input files into an output file suitable for spooling. Various options allow the user to specify input and output file units, title and banner generation, etc. CONCAT accepts commands either from a terminal or from a command file.

Using CONCAT

CONCAT is invoked from PRIMOS command level. Various options can be specified on the command line, as explained below. After processing the command line, CONCAT enters one of two modes. If **-COMMAND** is specified on the command line, command mode is entered and CONCAT accepts commands changing the operating environment (banners, titles, page numbering, disposition of files, etc.). Commands should be given one per line. If **-COMMAND** is not specified on the command line, insert mode is entered, and CONCAT accepts a list of input files, one file per line.

When CONCAT is in insert mode, it prints a colon prompt (:). To leave insert mode, the user follows the prompt with an empty line or carriage return. CONCAT responds by going into command mode and printing a right angle prompt (>). The user leaves command mode by typing QUIT.

Command line options

The following options may be given on the command line only. **outfile** (if specified) must be the first option; **-BANNER** (if specified) must be the last. Other options may be given in any order.

To specify output file:

outfile

A **filename** or pathname specifying the output file. If **outfile** is omitted, the file open on the output unit is used instead. (The default output unit is Unit 2, but this can be changed with the -OUNIT option.) If no file is open, and **outfile** is not specified, CONCAT returns an error message:

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

To specify file units:

-IUNIT n — Specifies the unit on which the input files will be opened. The default is 1.

-OUNIT n — Specifies the unit on which the output file will be opened. The default unit is 2. If **outfile** is omitted from the command line, the file open on unit **n** will be used for output.

To specify output file verification:

-VERIFY — If **outfile** already exists, causes an OK TO MODIFY OLD query followed by an OVERWRITE OR APPEND query. This is the default mode.

-OVERWRITE — Causes **outfile** to be overwritten if it already exists.

-APPEND — Causes output to be appended to **outfile** if it already exists.

To specify output file disposition:

-CLOSE — Truncates and closes **outfile** on exit. This is the default.

-OPEN — Leaves **outfile** open on exit. No truncation of **outfile** is done.

-TRUNCATE — On exit, truncates **outfile** but leaves it open.

To specify mode:

-INSERT — Goes directly into insert mode (prompt is ":") and accepts a list of files to be inserted into the output file. If neither -INSERT nor -COMMAND is specified on the command line, -INSERT is assumed.

Options/subcommands

The following instructions may be given as options on the command line (preceded by a hyphen), or as subcommands when CONCAT is in command mode. (For example, -HEADER is a command line option, while HEADER is a subcommand.)

Note

When CONCAT is in command mode, it ignores blank lines and text preceded by /*.

To specify input file disposition:

DELETE — Deletes input file after copying it to the output file. This option has no abbreviation.

NDELETE — Does not delete input file after copying it. This is the default.

To specify formatting:

HEADER — Specifies title generation and banner page suppression. This is the default. The first line of each input file is read. If it is a title (begins with octal 1 in the left byte), then the line only appears as the title for the file. Otherwise the line appears both as the title line and as the first line of the file.

BANNER [banner-line] Specifies title and banner page generation. It must be the last option on the line. A banner page is inserted between input files. It contains two lines of up to 14 large characters. The **banner-line** specifies the first of these lines. It is read as raw text so that spaces are accepted. The second line is the last component of the input file treename. Titles are generated as in -HEADER mode. The banner page has the same title as the routine following it. If the optional **banner-line** is omitted, then that line of the banner will be left blank.

NHEADER Suppresses both title and banner page generation. The input files are copied to the output file without modification.

EJECT Generates a page eject between input files. Suppresses title and banner page generation.

RESETP Resets spooler page numbering between input files.

NRESETP Does not reset spooler page numbering between input files. This is the default.

Subcommands

CONCAT recognizes three further subcommands:

TITLE [new-title] Use **new-title** as the header for the next input file. It is read as raw text so that spaces are accepted. If **new-title** is omitted the file name is used.

INSERT [file-name-list] If **file-name-list** is omitted, CONCAT will go into insert mode and accept the names of the files concatenated one per line. To exit from insert mode, a blank or null line may be entered. If **file-name-list** is specified, the files in the list are concatenated into the output file without entering insert mode. If an error is made in the line, the rest of the line after the error is ignored. Up to 40 files may be specified on one line, separated by spaces or commas. Treenames with imbedded spaces (i.e., passwords) must be enclosed in quotes.

QUIT Exit from CONCAT. This is the only clean way to exit from CONCAT.

▶ CONFIG ntusr pagedev comdev [maxpag] [altdev] [namlc] [nphan] [nrusr] [smlc]

Operator command

The CONFIG command defines system parameters. CONFIG is specified upon cold start of PRIMOS. It is an operator command, and it is issued at the supervisor terminal. For further information, refer to **The System Administrator's Guide.**

▶ COPY_DISK

Operator command

This operator command copies one physical disk, of any format, to another physical disk. COPY_DISK is a command for system operators, and is discussed in full detail in **The System Administrator's Guide.**

CPL filename

The CPL command invokes the CPL interpreter and executes a CPL program.

filename is the name of the file you want run. If **filename** ends in '.CPL', that file is run. Otherwise, CPL looks first for **filename . CPL,** and runs it, if found. If **filename . CPL** cannot be found, CPL looks for **filename**, running it if found.

For more information regarding CPL, see **The CPL User's Guide.**

CPMPC pathname [-CRn] [-PRINT]

CPMPC is a card-reader or punch utility command that causes a file specified by **pathname** to be punched onto a card deck. 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, refer to the discussion of CRMPC command.)

Parameters to the CPMPC command may be specified in any order. The value of the **-CRn** option is CR0 or CR1, depending on whether the first (CR0) or second (CR1) card-reader/punch is specified. Before invoking the CPMPC command, the card-reader/punch must be assigned by an ASSIGN CR0 or ASSIGN CR1.

The -PRINT option causes punched data to be interpreted (printed) on the card if the punch has that capability.

CREATE pathname

The CREATE command creates a new directory as specified by **pathname**. (If a simple directory name is given, the new directory is created subordinate to the current directory.) The owner password of the new directory is blank, and the nonowner password is zero (any password will match). Also the protection keys are set to 7 0.

```
OK, A ACCOUNTING SECRET
OK, CREATE SALES
OK, A *>SALES
OK< CR DOMESTIC
OK< CR INTERNATIONAL
OK, CR *>DOMESTIC>WESTERN
OK, CR *>DOMESTIC>CENTRAL
OK, CR *>DOMESTIC>EASTERN
```

CREATK

CREATK invokes a program to build a template for multiple keyed index files. It is a part of the MIDAS subsystem. When invoked, CREATK sets up a dialog that asks the user about the kinds of keyed index files that are to be created. For further information, refer to **The MIDAS Reference 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**. Reading 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 is pressed on the terminal, stopping CRMPC.

The parameters may be specified in any order. The option **-CRn** specifies the number of the card reader /punch. n may be 0 or 1.

If the **-PRINT** option is specified, the contents of each card are printed on the card (if the card reader has that capability).

▶ CSUBS

CSUBS, an external command, invokes the COBOL DBMS subschema. See **The DBMS COBOL Reference Guide** for details.

▶ CX { [pathname] [-PRIORITY n] [-CPULIMIT cpu-sec] / [option] }

See CX in **OLD COMMANDS, APPENDIX C.**

At Rev. 17, CX has been replaced by Batch environment job processing. The Batch environment accepts all existing CX files when these are submitted with the **JOB pathname** command.

▶ DATE [option]

DATE prints (or displays) the current date and time at the user's terminal. DATE is useful to date command output files.

option is one of the following:

Option	Sample Output
-FULL	8Ø-1Ø-13.1Ø:31:Ø4.Mon
-USA	1Ø/13/8Ø
-UFULL	1Ø/13/8Ø.1Ø:31:2Ø.Mon
-DAY	13
-MONTH	October
-YEAR	198Ø
-TIME	1Ø:32:16
-AMPM	1Ø:32 AM
-DOW	Monday
-CAL	October 13, 198Ø
-TAG	8Ø1Ø13
-FTAG	8Ø1Ø13.1Ø3328

If DATE is invoked with no option it prints:

```
Monday, October 13, 198Ø 1Ø:34 AM
```

▶ DBACP

Invokes data base administrator command process. Refer to **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

Invokes the source level debugger. A list of DBG subcommands is given in Appendix C. For a full discussion of debugging with DBG, see the **Source Level Debugger Reference Guide.**

▶ DBUTL

The DBUTL command invokes a data base dump utility that allows users to monitor the contents of a data base schema and shared user table. DBUTL accepts only uppercase commands. For details on its use, consult the **DBMS Schema Reference Guide**.

▶ DEFINE_GVAR pathname [-CREATE]

The DEVINE_GVAR command creates a file to contain global variables, or activates an existing global variable file. The form

DEFINE_GVAR pathname -CREATE

creates a new file where **pathname** is the name of the file you want created. **-CREATE** must be entered literally.

The command:

DEFINE_GVAR pathname

activates an existing file.

The DEFINE_GVAR command may be used at command level or inside a CPL program. You must create a global variable file before you define any global variables. 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. For example:

```
DEFINE_GVAR MY_VARS -CREATE
```

creates an empty global variable file named MY_VARS.

To use the file again in a later session, use the command:

```
DEFINE_GVAR MY_VARS
```

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

You may create more than one global variable file, but may only have one global variable file active at any time. Therefore, the DEFINE_GVAR command activates the named file and turns off any global variable file already active. Logging out also deactivates an active global variable file.

Note

If you have a password on the directory containing your global variable file, you must use a full pathname with the password when using the DEFINE_GVAR command. For example:

See also: **SET_VAR, LIST_VAR, DELETE_VAR**

▶ DELAY [min] [max] [margin]

The DELAY command defines a time function that delays the printing of a character after a Carriage Return (CR) has been output to a terminal. **min** defines the number of character-times (time it takes the system to type a character on a line) to delay when CR is output at the left margin. **max** defines the number of character-times to delay when CR is output at the right margin. **margin** defines the number of characters required to move to the right margin. If a CR is typed at some point within a line, the time delay is proportional to the number of characters typed. If **margin** is not specified, 72 is assumed; if **max** is not specified, 12 is assumed. If the command, DELAY, is given with no parameters, the default values 6, 12, and 72 are assumed; these values are adequate for most 30 cps terminals.

The DELAY command can be used prior to logging in.

Example:

```
DELAY Ø 1Ø 1ØØ
OK,
```

The DELAY command may be issued from the supervisor terminal if it is designated to be user 1 (refer to the USRASR command).

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

▶ DELETE pathname

DELETE frees the disk storage space used by the file or directory specified by **pathname** and removes the name from the specified or current UFD.

A directory cannot be deleted until all files within the directory have been deleted. If a directory is not empty, DELETE returns the message:

```
The directory is not empty.  DIRECTORY-NAME
ER!
```

To delete non-empty directories, use FUTIL's TREDEL or UFDDEL commands. To delete SEG runfiles, use the SEG DELETE subcommand.

▶ DELETE_VAR variable-names

The DELETE_VAR command deletes one or more global variables from an active global variable file. For example:

```
DEFINE_GVAR MY_VARS
DELETE_VAR .UFD
```

deletes the variable .UFD from the file MY_VARS. The command:

```
DELETE_VAR .A .B .C
```

deletes the three variables .A .B .C from the active file.

See also: **SET_VAR, LIST_VAR**

▶ DELSEG {segno / ALL}

DELSEG is an internal command that frees (deletes) segments.

The parameter **segno** is the segment number of the segment to be freed, **segno** must be '2000 or above for user 1 ('4001 and above for all other users) and not equal to '4000. Specifying **ALL** deletes all segments belonging to the user at that terminal. A "BAD PARAMETER" message is the response to an illegal segment number. Deleting an already nonexistent segment has no effect.

Example:

```
OK, DELSEG 4ØØ3
```

▶ DISKS [NOT] pdisk-0 [pdisk-1]...[pdisk-7] *Operator command*

The DISKS command adds or removes the specified physical disk(s) to/from the Assignable Disks Table.

▶ DMSTK [options]

The DMSTK command produces a call/return trace of the user's command loop stack and Static Mode stack (if any). Its **options**, which may be given in any order, specify how the dump is to be done. Addresses are always printed in octal.

Option	Meaning
-BRIEF	Specifies a short format dump, omitting condition frames and fault frames. If this option is not given, the dump will be done in full format. (Since "full format" is the default, there is no full format option.)
-ALL	Specifies that dumping is to begin with the frame from which DMSTK was called. If -ALL is not specified, dumping begins with the most recent condition frame (if there is one) or with the frame from which DMSTK was called.
-FROM n	Specifies that dumping is to begin from frame . (The frame from which DMSTK is called is frame 1.) If -FROM is not given, then the -ALL option determines the starting point for the dump.
FRAMES n	Specifies that only frames of the stack are to be dumped. (must be a positive decimal integer.) The default is to dump the entire stack.
-ON_UNITS	Specifies that a list of on-units established by each activation (i.e., frame) that is dumped is to be produced.

For information on the format of the stack dump, see Appendix D.

▶ DPTCFG

Operator command

The DPTCFG command constructs the configuration file needed to set up a DPTX system. For details, see the **Distributed Processing Terminal Executive Guide.**

▶ DPTX

Operator command

The System Administrator uses the DPTX command 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.**

▶ DROPDTR

The DROPDTR command forces the dropping of the DTR (Data Terminal Ready) signal associated with an AMLC line. As such, it is used only in the following situation:

1. A user has been talking to a Prime computer over a dial-up (AMLC) line, using a "port selector" or modem.
2. The user has logged out.
3. The user now wants to disconnect from the current line and re-connect to a new line. (For example, the user may wish to login to a different node on a network.) To force the disconnect, the logged-out user gives the DROPDTR command.

(Normally, DTR is dropped following a "grace period" of up to 10 minutes. The length of the grace period is set by the System Administrator, using the AMLTIM or DTRDRP configuration directives.)

▶ ED [pathname]

The ED command loads and starts EDITOR, the most commonly used version of the Prime text editor. If a **pathname** is specified, that file is loaded into EDITOR's text buffer in memory, and EDITOR is started in EDIT mode. Otherwise, the editor is started in INPUT mode with an empty text buffer. Files and units are automatically opened and closed. For details of ED operation, refer to **The New User's Guide to EDITOR and RUNOFF.**

Restarting Editor: If the user accidentally returns control to PRIMOS (for example, by a QUIT), the user can restart ED without losing any of the text buffer by issuing the command: START 1000 (continue from break) or START 1001 (resume in EDIT mode).

▶ EDB inputfile [outputfile]

The EDB command loads and starts EDB, the binary editor, which prints ENTER and waits for command input. EDB is used primarily for building and maintaining libraries of subroutines. The input and output files may be on disk or paper tape. If the pathname, **outputfile**, already exists, it is overwritten by the output file. If paper tape is used for either **inputfile** or **outputfile**, use the name **-PTR**. For example: EDB -PTR NEWLIB. For details, see **The PRIMOS Subroutines Reference Guide.**

▶ ELIGTS tenths *Operator command*

The ELIGTS command allows modification of the user-eligibility time slice. For details, see **The System Administrator's Guide**.

▶ F77 pathname [options]

The F77 command loads the Prime FORTRAN 77 compiler and compiles the object program from the ASCII file specified by **pathname**. If no conflicting options are specified, F77 will use the following default options:

```
-64V -OPTIMIZE -INTL -UPCASE -NOBIG -LOGL -L NO -B YES
```

The F77 compiler can be used to compile programs written in FORTRAN IV. The programs may be compiled in V mode or I mode. They must be loaded with SEG; F77 does not support R-mode compilation.

For further information on using the F77 compiler, see **The FORTRAN 77 Reference Guide.**

See also: **FTN**

▶ FAP

FAP invokes the FORMS Administrative Processor. For further information, see **The FORMS Programmer's Guide**.

▶ FDL

FDL invokes the FORMS Definition Language. For further information, see **The FORMS Programmer's Guide**.

▶ FDML

FDML invokes the FORTRAN DML preprocessor. See **The DBMS FORTRAN Reference Guide**.

▶ FILMEM [ALL]

Under PRIMOS, FILMEM with no argument fills memory locations '100 to the top of 32K with zeros. If running under PRIMOS II, FILMEM clears '100 to the top of 64K bytes, except for those locations occupied by PRIMOS II.

FILMEM ALL clears all of the user space (up to 128K bytes) (segment '4000).

▶ FILVER [pathname-1] [pathname-2]

FILVER invokes a file comparison and verification process for comparing runfiles.

When **pathname-1** and **pathname-2** are used, FILVER causes them to be compared for equivalence. If any differences exist, a message is printed indicating failure to verify. If the files are exactly the same, a message is printed that confirms successful verification. FILVER is an external command.

The differences are reported in a form useful for comparing runfiles. It is suggested that the user also have listings of both files compared to make use of the different information printed by FILVER. Four numbers are displayed for each difference:

DIFF wwwwww xxxxxx yyyyyy zzzzzz

where **wwwwww xxxxxx** describes the position of the file: wwwwww is a sector number (in octal); xxxxxx is the offset within the sector (in octal). The user must take into account the nine-word header in runfiles and any offset from a sector boundary in the starting location of the run file. The parameter **yyyyyy** is the value of the differing word in File 1, **zzzzzz** is the value of the word in File 2; both of these parameters are in octal.

See also: **CMPF**

Example:

```
OK, FILVER FILEA FILEB

DIFF 000000 000002 170765 171767
DIFF 000000 000003 164743 164746
DIFF 000000 000004 165612 172212
DIFF 000000 000005 161362 171345
DIFF 000000 000006 167767 162212
DIFF 000000 000007 167212 163357
DIFF 000000 000010 163357 174212
DIFF 000000 000011 174212 165365
DIFF 000000 000012 165365 166760
DIFF 000000 000013 166760 171612
DIFF 000000 000014 171612 167766
DIFF 000000 000015 167766 162762
DIFF 000000 000016 162762 105000
DIFF 000000 000017 105000 172350
DIFF 000000 000020 172350 162612
DIFF 000000 000021 162612 162357
DIFF 000000 000022 166341 163612
FILE LENGTHS DIFFER AS FOLLOWS:
FILE 1: 23 WORDS
FILE 2: 19 WORDS

  FILES NOT EQUAL
ER!
```

▶ FIXRAT [options] *Operator Command*

FIXRAT is a maintenance program that checks the file integrity of a disk pack. Under PRIMOS, the disk must be ASSIGNed; it cannot be running under PRIMOS by a STARTUP or ADDISK command. For a complete discussion of FIXRAT, with examples, see **The System Administrator's Guide**.

FSUBS

FSUBS invokes the FORTRAN DBMS subschema. See the **DBMS FORTRAN Reference Guide**.

FTN pathname [options]

The FTN command loads the Prime FORTRAN compiler and compiles the object program from an ASCII file, **pathname. The FORTRAN Reference Guide** gives a detailed discussion.

See also: **F77**

FUTIL

FUTIL invokes a file utility command that provides subsystem commands for the user to copy, delete, and list both files and directories. FUTIL also has an ATTACH command that allows attaching to subdirectories by giving a directory treename from either the MFD or home UFD to the specified subdirectory. FUTIL allows operations with files within UFDs and files within segment directories. FUTIL may be run from a command file.

The FUTIL prompt is the right-angle bracket (>). FUTIL accepts both upper and lowercase input; but password cases (lower or upper) must match.

The first time FUTIL sees a lowercase password, it prints the message:

Note

A password with lowercase characters will not be converted to uppercase by FUTIL.

See Section 4 for a detailed discussion of subsystem commands available under FUTIL.

HDXSTAT

The HDXSTAT command allows you to display 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 for the duration of the phone call only. For complete details, see The PRIMENET PTU.

The HDXSTAT command 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 will be one of the following:

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 up.
trying to establish	A line has been set to initiate calls to a remote site, but contact with the site has not been made.

HPSD

HPSD loads a version of PSD (Prime Symbolic Debugger) that is stored in the upper portions of memory. See **PMA Programmer's Guide** for details.

INPUT pathname

INPUT opens a source file on File Unit 1 for reading. The file is assigned the name **pathname**. If pathname is only a simple filename, it is in the current UFD. This command has the same effect

as OPEN pathname 1 1. (For PMA and FTN, the source file name is usually provided with the command that starts assembly or compilation.) INPUT is an internal command.

▶ **JOB** { **filename** / **job-id** } **[options]**

The JOB command controls individual jobs run in the Batch subsystem. It has three major groups of options that allow users to submit, monitor and control their jobs, and a fourth group that provides operator control for waiting or executing jobs.

Submitting a job

The format and options for submitting a job are:

JOB filename [**-ACCT information** / **-ARGS cpl-arguments** / **-CPL** / **-CPTIME seconds** / **-ETIME minutes** / **-FUNIT number** / **-HOME pathname** / **-PRIORITY value** / **-QUEUE queuename** / **-RESTART** { **YES** / **NO** }]

If the simplest form, **JOB filename**, is used, the Batch monitor looks for a file in the user's working directory. It looks first for **filename.CPL**, then for **filename**. (Batch assumes that all files whose names end in .CPL are CPL files, and that all other files are command files.)

If it finds either **filename.CPL** or **filename**, the monitor gives the job the user's login name, places it in an appropriate queue, assigns it a **job-id**, **#snnnn** (where **s** is the number of the queue, and **nnnn** is the jobs number within the queue), and fills in the remaining options with the chosen queue's default values.

The **options** themselves have the following values and meanings:

Note

All numbers must be given as decimal integers.

Option	Description
-ACCT information	Allows the user to specify accounting **information** for his job. The **information** must be 80 characters or less in length. It may not be an explicit register setting or be preceded by an unquoted minus sign. If it contains spaces, commas, or comment designators (/*), it should be enclosed in apostrophes. The **information** will be included in job DISPLAYs, but will otherwise be ignored by Batch.
-ARGS cpl-arguments	Used to pass CPL arguments to the job being processed. **-ARGS** must be the last option issued on a command line, as everything that follows the **-ARGS** option on the command line (except comments) is assumed to be the CPL arguments being passed. JOB doesn't read the CPL arguments; it just passes them to the CPL file when execution of the file begins.

-CPL

Runs submitted file as a CPL file, no matter what the file's name is.

-CPTIME { **seconds** / **NONE** }

Specifies the maximum amount of CPU time (in **seconds**) to be allotted to the job. **NONE** requests that no time limit be placed on the job. If the job exceeds the time limit, it will be aborted. If the CPTIME specification conflicts with the CPTIME limit of the requested queue, an error message will ask the user to specify a new limit. If it conflicts with all queues (and no queue name was specified) the error message "no queue available for job" will be printed.

-ETIME { **minutes** / **NONE** }

Specifies (in **minutes**) the elapsed time to be allowed before the job is aborted. Details are the same as those for -CPTIME.

-FUNIT number

This option is used for command input files only. It *cannot* be used for CPL programs. When used, it specifies the file unit to be used for command input. Permissible values are from 1 to 126, unless the System Administrator has set a lower limit. (Smallest possible range is 1 to 16.) Default depends on queue parameters; it is usually 6. (CPL jobs have file units allocated dynamically.)

-HOME pathname

Specifies the UFD in which a job is to run. Using this option has the same effect as providing an ATTACH command as the first line of the command file. The **pathname** for a -HOME option, however, may not be a null specification or a relative pathname (i.e., it may not begin with *>), and may not exceed 80 characters in length.

-PRIORITY value

Determines the job's priority within its queue. Possible **values** are from 0 to 9, with 9 being the highest (most favored) priority. Default is queue-dependent.

-QUEUE queuename

Allows the user to specify the name of the queue in which his job should be placed. (To learn the names and characteristics of queues, use the BATGEN -DISPLAY command.) If a conflict results between -QUEUE and some other option (e.g., if a user specifies -CPTIME NONE -QUEUE ALPHA, when queue ALPHA allows a maximum of 120 seconds CPU time), an error message will request the user to resubmit the job with different options.

-RESTART { **YES** / **NO** }

Determines whether a job can or cannot be restarted following a -RESTART command or a system shutdown. The default is always -RESTART YES.

Note

Any or all of the above options may be specified within the command input file itself by using a **$$ JOB command** as the first non-comment line of the file. The command format is:

$$ JOB user-name [options]

where user-name is either * or the name the user logged in with, and options represents the eight JOB options listed above. If user-name is * , any user may submit the file to Batch; otherwise, only people logged in as user-name can submit it.

Example:

```
$$ JOB JONES -HOME JONES>REPORTS -CPTIME NONE -ETIME NONE
```

Any job may specify options via the JOB command, the $$ JOB command, or both. If one option is specified twice, the JOB option overrules the $$ JOB option. For example, if $$ JOB called for -CPTIME 20 and JOB called for -CPTIME 10, the job would be allowed only 10 seconds of CPU time before being aborted.

Monitoring the job

While a job is in a Batch queue, its progress can be monitored by the JOB command form:

JOB [**job-id** / **jobname**] { **-STATUS** / **-DISPLAY** }

Within this format, the **-STATUS** and **-DISPLAY** options govern the amount of information to be shown, while the **jobname** and **job-id** options allow the user to specify the jobs on which he wants information, as follows:

Option	Description
job-id	A 5-digit number preceded by a #, assigned to a job by the monitor when the job is placed in a queue. Use the **job-id** to request information on one job only.
jobname	The name of the file being run. If the job was submitted as a pathname (e.g., JOB FELLOWSHIP>HOBBITS>FRODO), its **job-name** is the final element of the pathname (e.g., FRODO). Use this format to request information on multiple submissions of a file.

Note

Omitting **jobname** and **job-id** requests information on all the user's jobs.

-STATUS	Prints out the job's **jobname** and **job-id**, the name of the queue in which it is placed, and its execution status: whether it is held, waiting, or running.
-DISPLAY	Provides status information and values for all JOB and $$ JOB command options (except -HOME) — both those specified by the user and those assumed from queue-defined defaults.

JOB CONTROL

The following forms of the JOB command may be used to control the job's execution or characteristics once it is in the queue.

JOB job-id { **-ABORT** / **-CHANGE [options]** / **-CANCEL** / **-RESTART** }

Note

Jobname may be used in place of **job-id** only if the user has only one active job of that name.

Option	Description
-CANCEL	Prevents the execution of jobs waiting to run. -CANCELing a running job will not halt its execution. However, it will make it impossible to RESTART that particular job, unless -CHANGE -RESTART YES is performed.
-ABORT	Terminates the execution of a running job and CANCELs a held or waiting job.
-CHANGE	Allows the user to change the -ACCT, -ARGS, -CPTIME, -ETIME, -FUNIT, -HOME, or -RESTART options on a waiting job. -CPL, -QUEUE and -PRIORITY cannot be changed.

If the job is already running, changes can be made by invoking first the JOB -CHANGE command and then the JOB -RESTART command.

Example:

```
JOB TEST4 -CHANGE -CPTIME 12Ø -ACCT MARKETING -RESTART YES
JOB TEST4 -RESTART
```

The job will abort, then restart under the newly specified options.

Operator commands

The operator's form of the job command is:

JOB job-id { **-ABORT** | **-CANCEL** | **-HOLD** | **-RELEASE** | **-RESTART** }

With the **-HOLD** option, the operator can hold a waiting job in the queue, keeping it active but preventing it from running until he **RELEASE**s it. Users with jobs that need unavailable peripheral equipment, or jobs that should run after a particular time may ask the operator to HOLD the jobs until the equipment is available or the time is right.

The operator's **-ABORT, -CANCEL**, and **-RESTART** options are identical to the user options with those names.

▶ KBUILD

KBUILD builds a keyed-index file of fixed-length records. It is part of the MIDAS subsystem. When invoked, KBUILD asks the user a series of questions about the file to be built. For further information, refer to **The MIDAS Reference Guide.**

▶ KIDDEL

KIDDEL invokes a program to delete all or part of the records in a keyed index file. It is part of the MIDAS subsystem. For further information, refer to **The MIDAS Reference Guide.**

▶ **LABEL MTn [-TYPE type]** [{ **-VOLSER** | **-VOLID** | **-VOLUME** } **volume-id]** **[-OWNER owner]**
[-ACCESS access] **[-INIT]**

LABEL initializes magnetic tapes just as MAKE initializes disks. LABEL writes either IBM (9-track EBCDIC or 7-track BCD) or ANSI (9-track ASCII) level 1 volume labels followed by dummy HDR1 and EOF1 labels. LABEL can also be used to read existing VOL1 and HDR1 labels. ANSI labels are written in accordance with the American National Standards Institute standard ANSI X3.27-1978. IBM labels are written in accordance with IBM's specifications (IBM manual GC28-6680-5). Any non-standard labels such as 7-track ASCII or user-defined labels *cannot* be read or written.

Using LABEL

To read existing labels type the command:

LABEL MTn [-TYPE type]

To write labels type the command:

LABEL MTn [-TYPE type] -VOLID volume-id [-OWNER owner] [-ACCESS access] [-INIT]

MTn The tape drive where the tape to be labelled is located. **n** is a number between 0 and 7. This keyword is required and must be the first on the command line.

type
-TYPE A = 9-track ASCII (ANSI) (this is the default)
-TYPE B = 7-track BCD (IBM)
-TYPE E = 9-track EBCDIC (IBM)

volume-id A 1-6 character string which uniquely identifies this tape reel. If less than 6 characters are specified, they are blank-padded on the right. The keywords -VOLUME or -VOL may be substituted for the keyword -VOLID.

owner 1-14 characters long for ANSI labels, 1-10 characters long for IBM labels. If less than 14 (or 10) characters are specified, they are blank-padded on the right. If this keyword is omitted, the default is the user's login name. The keyword -OWN may be substituted for the keyword -OWNER.

access A single character defining access to this tape. ACCESS is not used by Prime software but is included for completeness. If it is omitted it is left blank on ANSI labels. ACCESS is ignored for IBM labels.

-INIT Necessary keyword for previously unformatted tapes.

ERRORS USING LABEL

Improper use of the LABEL command causes an error message to be printed. These errors are the result of bad syntax in the LABEL command itself or a system magnetic tape I/O error.

Syntax errors

▶ *****DUPLICATE KEYWORD DETECTED**

The same keyword was typed more than once. (Error 1)

▶ *****INVALID TAPE UNIT SPECIFIED**

Something other than MT0-MT7 was typed. (Error 2)

▶ *****VOLUME ID SPECIFIED IS TOO LONG**

The volume id cannot be longer than six characters. (Error 3)

▶ *****OWNER ID SPECIFIED IS TOO LONG**

The owner id cannot be longer than 14 characters. (Error 4)

▶ *****INVALID LABEL TYPE SPECIFIED**

Label type must be one of the characters "A", "E", or "B". (Error 5)

▶ *****NO MAGNETIC TAPE UNIT SPECIFIED**

A magnetic tape unit is required. (Error 6)

▶ *****VOLUME ID WAS NOT SPECIFIED**

When writing labels, a volume id is required. (Error 7)

▶ **OWNER ID SPECIFIED IS TOO LONG FOR TYPES B OR E**

The owner id for IBM labels cannot be longer than 10 characters. (Error 8)

▶ *****UNABLE TO WRITE TAPE LABEL ON THIS TAPE**

A mag tape read error occurred and the label was not written. (Error 9)

▶ *****UNABLE TO READ TAPE LABEL ON THIS TAPE**

A mag tape read error occurred and the label was not read. (Error 10)

▶ *****LABEL OPERATION ABORTED**

Error 9, 10, 12, or 14 occurred and the label was not read. (Error 11)

▶ *****LABEL READ WAS NOT TYPE x**

The label read was not of the type specified. (Error 12)

▶ *****ACCESS IGNORED FOR IBM LABELS (WARNING ONLY)**

This is a warning only — processing continues. (Error 13)

▶ *****VOL1 LABEL ALREADY EXISTS**

ANSI standards prohibit the rewriting of VOL1 labels. (Error 14)

▶ **UNRECOGNIZED KEYWORD. string (CMDL$A)**

An invalid keyword (string) appeared on the command line. (Error 15)

System errors

Error	Meaning
MTn NOT ASSIGNED	Use command AS MTn before the LABEL command
subr EOF	END-OF-FILE on the magnetic tape
subr EOT	END-OF-TAPE
subr MTNO	The tape drive is not operational
subr PERR	Parity error on the tape drive
subr HERR	Tape drive hardware error
subr BADC	LABEL improperly called mag tape subroutines

(In the above errors, **subr** is the name of the mag tape subroutine that reported the error. See **PRIMOS Subroutines Reference Guide**, for more information regarding these errors.)

If LABEL successfully writes a label, the message "TAPE LABEL WAS WRITTEN SUCCESSFULLY" is displayed. On read operations, LABEL prints out the volume and owner ids, creation date, access (ANSI tapes only), and other information.

HELP FACILITY

The command LABEL -HELP causes LABEL to print out a description of the command similar to that found in this document.

For a complete description of tape labels and their use, refer to the IBM publication GC28-6680, OS TAPE LABELS and the ANSI publication X3.27-1969, "American National Standard Magnetic Tape Labels for Information Interchange".

▶ LATE

LATE requests the time at which the next command is to be accepted. The LATE command responds as follows:

```
Time of day (HHMM) to execute next command:
```

The user then types in the time of day. The time of the next command is expressed as a number of the form **HHMM. HH** is the hour (00 through 23), and **MM** is the minute (00 through 59). LATE responds to this input with the message:

```
Next command will be executed at HH:MM hours.
```

If the specified time is earlier than the current time, execution of the command is deferred until the following day. For example:

```
OK, late
[LATE rev 18.Ø]

Time of day (HHMM) to execute next command: Ø23Ø

Next command will be executed at Ø2:3Ø tomorrow.

OK, late
[LATE rev 18.Ø]

Time of day (HHMM) to execute next command: 2245

Next command will be executed at 22:45.
```

If the colon is omitted from the time specification, LATE assumes the last two digits designate minutes. For example, 30 is interpreted as 00 hours and 30 minutes. If no time specification is made, the system defaults to 00 (midnight).

If the colon is included, the first two digits designate hours and the last two minutes.

No other commands can be executed before the specified time except a CONTROL-P to abort from LATE. LATE is useful if a user wishes to defer execution of a process, such as a command file, until a time when it is expected that system load will be light, such as during the second shift.

▶ LISTF

LISTF prints the current UFD pathname (if the pathname is less than 80 characters long), the logical device upon which the UFD resides, and all filenames in the UFD. Attributes of files such as type, size, and protection may be examined using the LISTF subcommand of the FUTIL command.

LISTF prints the word OWNER following the device number upon which the UFD resides, if the user is an owner. If the user is a nonowner, the LISTF command prints the word NONOWN following the device number. The concept of owner and nonowner is described in the **PRIMOS Subroutines Reference Guide** under File Access, and is associated with the commands PASSWD and PROTEC.

Example:

```
OK, listf

UFD=<FOREST>BEECH>BRANCH5     5  OWNER

REPORT  TMP     KELLY   UNICORNS        OLD.STUFF       COOKIE.MONSTER
CLASS.NOTES

OK,
```

▶ LISTING pathname

LISTING opens a file for writing on File Unit 2, usually as a listing output file for a compiler or assembler. All subsequent compilation and assembly listings go to this file until it is closed. The **pathname** specifies the directory and filename of the listing file. If **pathname** is only a simple filename, it is in the current UFD. LISTING is an internal command, and has the same effect as OPEN pathname 2 2.

Note

If no LISTING command has been given, PMA and COBOL automatically put listings into files named source-filename . LIST or L_source-filename.

▶ LIST_VAR [variable-name [...variable-name]]

THE LIST_VAR command lists all global variables contained in an active global variable file.

If the LIST_VAR command is given variable names as arguments, it lists only those variables. The arguments may be expressed as wildcard names. In this case, those variables whose names match the wildcards will be listed. For example:

```
OK, list_var
 .AWAY                          BEECH>BRANCH2>TWIG4
 .HOME                          BEECH>BRANCH5>TWIG3
 OK,

 OK, list_var .away
 .AWAY                          BEECH>BRANCH2>TWIG4
```

▶ LOAD

This command loads and starts LOAD, Prime's Linking Loader for PRIMOS. For a complete discussion of the loader and an example of the use of LOAD, refer to the **LOAD and SEG Reference Guide.**

LOAD loads programs for R-mode code generated by PMA, FORTRAN, or RPGII. To load segmented code, use the command SEG.

▶ LOGIN ufdname [password] [ldisk] [-ON nodename]

LOGIN is the command the user must type at the terminal to begin an operating session.

ufdname must be a valid UFD name on any of the disks available to the system, **password** is an optional argument that specifies the owner password, and **ldisk** is an optional argument that specifies logical device numbers to be searched for **ufdname**.

If the UFD has an owner **password**, the user must supply it at LOGIN time.

When LOGIN is successful, the user is attached to the UFD specified by **ufdname**. A login message is printed at the terminal and at the supervisor terminal. For example:

```
LOGIN JHNDOE

JHNDOE (2) LOGGED IN AT 12:39   Ø1Ø48Ø
```

The number in parentheses is the user number of the user terminal. The next number is the time of day, and the last number is the date in the form: **mmddyy**.

```
LOGIN JHNDOE GEMINI
```

logs in the user and attaches the UFD, JHNDOE, if the password GEMINI is correct.

Once logged in, users may attach to different UFDs. However, PRIMOS still identifies them by their original login names. Thus, if a user logs in as JHNDOE, he appears as JHNDOE on spool lists, status queries, phantoms, and logout messages, no matter what UFD he is attached to at the time.

Remote login: If the user's system is connected via a network to other PRIMOS systems, it is possible to log into a UFD that is not local to the system to which the user terminal is configured. This is accomplished by use of the **-ON** option as follows:

LOGIN ufdname -ON nodename

The parameter **nodename** is the name of the system in the network PRIMENET on which the UFD specified by **ufdname** resides. The number of remote users allowed to log into a given remote system is determined by the CONFIG command on the remote system. If no remote users have been specified in a given system's CONFIG command line, the **-ON** option with reference to that system's **nodename** is disabled. An example of a remote login is:

```
LOGIN HOBBIT -ON SYSD
```

If -ON option is omitted, an attempt is made to log into **ufdname** on the local system. If **-ON nodename** is supplied, the terminal is connected via PRIMENET to a process on node **nodename**, if it is available. If **nodename** is the name of the local node, the login attempt is made locally without the use of PRIMENET.

If the LOGIN command fails for any reason (e.g., NOT FOUND, INSUFFICIENT ACCESS RIGHTS, NO RIGHT), the PRIMENET connection is broken, and the terminal reconnected to a local (not logged-in) process.

Login messages: In addition to normal LOGIN errors, the following errors can occur.

LINE TO REMOTE SYSTEM DOWN

The PRIMENET link to **nodename** used by Remote Login is not operational. Remote login is restricted to using a single type of PRIMENET link at a time. Therefore, this message is possible even if an alternate link to nodename is operational. See the RLOGIN configuration statement in **The System Administrator's Guide.**

INVALID SYSTEM NAME

The system name specified in the LOGIN command is not defined.

NO MORE REMOTE TTY(S)

The maximum number of local terminals allowed to login to remote systems has been reached. Try again at a later time.

REMOTE SYSTEM HAS NO FREE REMOTE USERS

The maximum number of processes configured for use by remote terminals has been reached on the indicated remote system. Try again at a later time.

LOGIN NOT ALLOWED TO REMOTE SYSTEM

Local system configuration parameters have disallowed remote login to the specified system.

LOGIN NOT SUPPORTED TO REMOTE SYSTEM

The specified remote system does not support remote login.

BAD LOGIN

Other syntax or spelling error in the LOGIN command, or malfunction.

MAXUSR EXCEEDED

The maximum number of users allowed by the system administrator are already using PRIMOS. Try again later, or see your System Administrator.

LOGOUT PLEASE

The user already logged in has input a subsequent LOGIN command to PRIMOS without previously logging out, and implicit login is not implemented on that computer configuration; or, the user is trying to login to a different system than the one to which she is currently logged in.

▶ LOGOUT [-uu]

LOGOUT is the last command the user issues when giving up access to the system.

During LOGOUT, all user files are closed, all devices ASSIGNed to the user's terminal are released, the UFD is detached, and a logout message is printed at the user's terminal and at the supervisor terminal. All segments that were used by the user are returned to the supervisor when the user logs out.

If a user number is specified (**-uu**), that user (which may be a phantom or a process on another terminal) will be logged out, provided that user **-uu** has the same login name as the user giving the command. (At the system terminal, the user specified by **-uu** will be logged out, *regardless* of login name.)

Example:

```
LO
```

A typical response at the user terminal and also at the supervisor terminal might be as follows:

```
JHNDOE (2) LOGGED OUT AT 13:16 Ø30480
TIME USED= Ø:37  3:Ø1  Ø:54
```

The first number after 'TIME USED =' is the connect time in hours and minutes; the second number is CPU time in minutes and seconds, and the third number is disk I/O time in minutes and seconds. For more details about time accounting, refer to the TIME command.

If a user is logged out because the inactivity time has run out, the message TIMEOUT is printed at both user terminal and the supervisor terminal, followed by the normal logout message. If the user is logged out by a command issued at the supervisor terminal, the message FORCED LOGOUT is printed at both the user terminal and the supervisor terminal, followed by the normal logout message.

▶ LOGPRT [destination] [option(s)] *Operator command*

LOGPRT writes the contents of either LOGREC (the system event log) or NETREC (the network event log) into a disk file or displays them at a terminal. For a detailed discussion of event logging, see **The System Administrator's Guide**.

▶ LOOK [-uu] [segno] [access] [mapseg] *Operator command*

LOOK is an operator command that provides access to any segment in the system. This command is issued at the supervisor terminal, and it must be preceded by an OPRPRI 1 command. LOOK is discussed in **The System Administrator's Guide**.

▶ MAGNET

MAGNET is an interactive magnetic tape utility designed to simplify the transfer of non-Prime format magnetic tapes to and from the Prime computer. Its capabilities are as follows:

- Reads, writes, positions, and copies both 7- and 9-track tapes
- Supports 556-, 800-, and 1600-bpi tapes, depending on transport
- Performs record blocking/unblocking
- Provides ASCII, BINARY, EBCDIC, and BCD translation
- Maximum of 10,000 bytes per disk or tape record (2048 bytes for PRIMOS III)
- Fixed length records, unlabeled tapes only.

MAGNET is an external command. When invoked, MAGNET prints a release number and a date and then requests an option. For example:

```
OK, magnet

[MAGNET rev. 17.4]

OPTION:
```

The user may then issue one of the following commands:

Command	Function
POSITION	Positions the tape to a file/record
READ	Reads a file from tape to disk
WRITE	Writes a file to tape from disk
COPY	Copies a file from one tape to another
QUIT	Returns to PRIMOS.

All the MAGNET commands may be abbreviated to the first two characters. A detailed discussion of each MAGNET command is given in the following paragraphs.

POSITION command: The POSITION command positions the magnetic tape to a specified file and record number. Position has two modes: **absolute** and **relative**. An **absolute** position causes the tape to be rewound before any spacing occurs. A **relative** position allows the tape to be moved forward or backward from the current position.

When the POSITION command is given, MAGNET requests a magnetic tape unit number. The response is an integer in the range 0 to 7, optionally followed by a /7 or /9 to indicate a 7- or 9-track transport; if omitted, 9-track is assumed. MAGNET then asks whether this is an absolute or relative position. The first character of the response must be either A or R; otherwise, the question is repeated. Finally, the file and record numbers are requested. If an absolute position has been specified, the file and record numbers are absolute (starting with 1) and must be positive. If relative mode, the file number represents the number of files to forward-space or backspace and may be positive or negative. The record number is the number of records to space forward or backward and must be greater than or equal to zero.

An example of a typical POSITION operation is:

```
OK, as mtØ
Device MTØ assigned.
OK, magnet

[MAGNET rev. 17.4]

OPTION: position
MTU # = Ø/9
RELATIVE OR ABSOLUTE? a
FILE # = 3
RECORD # = 1
DONE.
OK,
```

READ command: The READ command allows a file to be read from the magnetic tape and written to disk, optionally providing unblocking and EBCDIC or BCD translation.

The maximum tape record size allowed by the READ command is 10,000 bytes. For 7-track non-BCD tapes, the buffer size is reduced by 1/3 because of the required 6-6-4 unpacking. The logical record (line image) size is 2048 bytes in all cases.

The READ command requests the unit number, which must be entered in the format described under the POSITION command. The user must then enter the magnetic tape file number as a positive integer. If greater than 0, the tape is rewound and positioned to the specified file number. If 0, no tape positioning occurs.

The next series of questions asked by the READ command relates to the format of the data on the magnetic tape file to be read. Logical record size is the number of bytes in a line image and must be no more than 2048. Blocking factor is the number of logical records (line images) contained in one tape record. MAGNET then asks the type of translation to be provided and requires one of the following responses:

ASCII	Indicates that no translation is to occur between tape and disk. The data is written to the disk file in ASCII format.
EBCDIC	Indicates that the data on the tape is to be translated from EBCDIC to ASCII before being written to the disk file.
BCD	Specifies that the data is to be translated from BCD (6-bit) to ASCII before being written to the disk file. This option is only meaningful when used with a 7-track tape. No 6-6-4 unpacking is done by MAGNET when this option is specified.
BINARY	Indicates that the data is to be written verbatim to a binary disk file. The record size is the specified logical record size. No translation occurs.

If either EBCDIC or BCD translation is specified, MAGNET asks if the entire record is to be translated or only the selected fields. This permits bypassing translation of binary fields in EBCDIC records (output, for example, by COBOL). For partial record translation, MAGNET requests starting and ending column numbers of the data to be translated. This input is terminated with a null line (CR only). Finally, MAGNET requests the disk output file name. If the named file already exists, MAGNET so states and asks if the file may be overwritten. If not, another output filename is requested. When the file has been read, a message is printed on the user terminal containing the number of tape records processed and the number of line images written.

An example of a typical interaction is:

```
OK, magnet

[MAGNET rev. 17.4]

OPTION: read
MTU # = Ø
MT FILE # = 1
LOGICAL RECORD SIZE = 8Ø
BLOCKING FACTOR = 1
ASCII, BCD, BINARY, OR EBCDIC? ebcdic
FULL OR PARTIAL RECORD TRANSLATION? partial

ENTER PAIRS OF STARTING/ENDING COLUMN NUMBERS, ONE PAIR PER LINE;
SEPERATE EACH COLUMN NUMBER WITH A COMMA.  TERMINATE ENTRY WITH
A NULL LINE (RETURN ONLY).

: 1,4Ø
: 43,8Ø
:

OUTPUT FILE: tap.Ø1

DONE...          14 PHYSICAL RECORDS READ,          14 LINES OUTPUT
OK,
```

When either READ or WRITE comes to the physical end-of-tape marker, the message

```
MT EOT

MOUNTING ANOTHER TAPE?
```

appears. If the answer is to be YES, *first* mount the new tape, *then* give the answer.

WRITE command: The WRITE command is similar to the READ command except that the file on disk is written to the magnetic tape. WRITE also provides facilities for blocking and character translation.

The WRITE command asks about the magnetic tape unit and file numbers, and then about the data format as it is to appear on the magnetic tape. The logical record size and blocking factor must be specified, as described for READ. The type of translation to be provided, if any, must also be input as follows: **ASCII** specifies no translation; **EBCDIC** and **BCD** specify appropriate translation onto tape; and **BINARY** specifies that the input file is a binary (FORTRAN unformatted) data file and that no translation is to be provided. If EBCDIC or BCD translation is specified, MAGNET asks whether the entire record or just certain fields are to be translated; see READ for description. Finally, MAGNET requests the name of the input disk file.

An example of a typical WRITE interaction is:

```
OK, magnet

[MAGNET rev. 17.4]

OPTION: write
MTU # = Ø
MT FILE # = 1
LOGICAL RECORD SIZE = 12Ø
BLOCKING FACTOR = 1
ASCII, BCD, BINARY, OR EBCDIC? ascii
INPUT FILE: tap.Ø5

DONE...          14 PHYSICAL RECORDS OUTPUT TO TAPE
OK,
```

COPY command: The COPY command copies a file (or files) from one magnetic tape to another. No character translation is provided for this operation. Either magnetic tape transport may be 7- or 9-track.

MAGNET requests the **FROM** and **TO** tape units, starting file numbers, and the number of files to copy. To copy an entire tape, the user must enter a large number (EOT is detected and causes the copy operation to halt).

An example of a typical COPY interaction is:

```
OK, as mtØ
Device MTØ assigned.
OK, as mt1
Device MT1 assigned.
OK, magnet

[MAGNET rev. 17.4]

OPTION: copy
'FROM' TAPE:
MTU # = 1
STARTING FILE # = 1

'TO' TAPE:
MTU # = Ø
STARTING FILE # = 2

# FILES TO COPY = 1
PRINT RECORD SIZES? y


FILE #     1
 RECORD          1 =    4Ø WORDS
 ---EOF---

DONE.
OK,
```

▶ MAGRST [-7TRK] [-TTY]

MAGRST restores information from a magnetic tape created by MAGSAV into the PRIMOS system. When invoked, MAGRST responds with a series of questions. These questions and appropriate user replies are discussed after the paragraphs that describe general information and the **-7TRK** and **-TTY** options.

General information: MAGSAV and MAGRST are utility programs that move files on any disk including the storage module, to a 7- or 9-track magnetic tape and vice versa. The files may be SAM, DAM, segment directories, UFDs, or an entire disk. Whenever a directory is specified, the directory and all components (the subtree) are transferred.

Logical Tapes: A logical tape consists of a header record, a file mark, file records, and two file marks. A logical tape may span multiple physical tapes, or a single physical tape may contain multiple logical tapes. The header record contains the tape name, date, and revision number.

Pathnames: A disk file appears on tape as a record containing a **pathname**, followed by as many data records as are required for the file. The **pathname** contains the unique path to the file specified by the user relative to the current logical volume and/or directory. When an entire disk is saved, all **pathnames** begin in the MFD. For example, a file might have a **pathname** of MFD>UFD>JUNK or MFD>UFD>SUBUFD>JUNK. MAGRST's $A (ATTACH) command accepts *only* filenames; it does not accept pathnames. Therefore, it is simplest to attach to the UFD into which you wish to restore the tape before invoking MAGRST.

Note

An attempt to restore a file called MFD while attached to an MFD results in the message:

```
FILETYPE MISMATCH, FILE OMITTED: MFD
```

The file is not restored, therefore.

Use of MAGRST: If the option **-7TRK** is specified, it indicates to MAGRST to use the 7-track tape format. The default value is 9-track. All restore operations take place in the home UFD. MAGRST asks for the tape unit and logical tape number. If the **-TTY** option is specified, MAGRST takes the unit number from the terminal, but takes all its other information from its current input stream. (This might be a command file or a CPL file.) The tape name, the date, and the revision are printed on the user terminal. Then MAGRST asks:

READY TO RESTORE:

The responses are as follows:

Response	Meaning
YES	Restore the entire tape.
NO	Don't restore; request a different tape unit and logical tape.
PARTIAL	Permits a restore of part of the tape.
NW [filename] [level]	Creates an index of the magnetic tape but does not write the tape to disk. **level** indicates the level of the index (e.g., 3). If **level** is not specified, the default level is 100. **filename** is the name of a file that will contain the index. If **filename** is not specified the index is printed (or displayed) at the terminal.
$I [filename] [level]	Causes an index to be printed when a YES (entire) or PARTIAL restore is done. If **level** is not specified, the default level is 2.

$A directory [passwd] [ldisk] key

Attach to a directory. **1disk** must be specified if it is not zero. (The STATUS DISK command tells what the ldisk numbers are.)

To attach a subdirectory, first attach the top directory, then use a key of 2 to attach downward.

For example:

```
$A UFD 6
$A SUBUFD 6 2
```

Attaching with a pathname is not supported. The syntax of $A is described in greater detail in Appendix C.

MAGRST provides the ability to enter multiple pathnames for a partial restore. For example, pathnames might be input in response to the query TREE NAME as follows:

```
TREE NAME:MFD>LIB>FTNLIB
TREE NAME:MFD>LIB>JUNK
TREE NAME:MFD>CMDNCO>PRINT
TREE NAME:
```

After each file is restored, the message:

```
FILE COMPLETE
```

is printed at the teriminal, if the $I command was specified. For a partial restore, files that have bad records are omitted. The pathnames of these files are printed along with an error message. The message:

```
RESTORE COMPLETE
```

is printed when the end of logical tape is reached.

MAGRST checks for conflicting file types when a file is going to be overwritten. Conflicts generate an error message, and the file is skipped.

It is not necessary for MAGRST to read through all logical tapes when restoring sequential logical tapes. After MAGRST has exited to PRIMOS, the magnetic tape is not rewound. Instead, it is positioned at the location before the beginning of the next logical tape in sequence. In the case of sequential logical tapes, the user must run MAGRST again and specify 0 for logical tape number where LOGICAL TAPE NO: is requested. Then, the next logical tape is restored without rewinding and reading through the preceding logical tapes.

Index: MAGRST allows a user to index a tape and direct the listing of the index to a disk file rather than the user terminal. To use this feature, follow the NW or the I command with a filename, and then with the number of **index** levels.

Example of index with MAGRST:

```
READY TO RESTORE: NW TAPE#1  5
```

In this case, MAGRST writes an **index** to level 5 into the file specified by TAPE#1.

Physical end of tape: When physical END OF TAPE is encountered in either MAGSAV or MAGRST, a message is logged on the user terminal and a new tape unit is requested. The new unit may be the same as the old unit.

Errors: Tape read or write errors are retried and tested for various conditions until considered unrecoverable. The first record on a tape is not retried. Both recovered and unrecovered errors are logged. If recoverable tape I/O errors occur, a total of the errors is printed at the end of the logical or physical tape. (e.g., 5 RECOVERED MT I/O ERRORS)

Unrecoverable errors cause the message to be printed.

```
MT READ ER UNIT STATUS SEQNO UNRECOVERED
```

UNIT	Tape drive unit number
STATUS	Status word from the read request
SEQNO	Sequence number of the invalid record

The next message will give the sequence number of the next good record read.

```
MT READ ER UNIT STATUS SEQNO RECOVERED
```

Assigning tapes: When running MAGRST under PRIMOS, the magnetic tape drive must be assigned. Refer to the description of the ASSIGN command for further details. Users must avoid restoring files into UFDs that are in use by other users because this action could either confuse the other users or could cause either MAGRST or the user program to abort with the message:

```
FILE IN USE
```

In this case, the abort is caused by two different programs attempting to gain access to the same file at the same time.

The following example shows a MAGRST of the tape made by MAGSAV in the example in the following section:

```
OK, magrst
REV. 17.4
YOU ARE NOT ATTACHED TO AN MFD
TAPE UNIT (9 TRK): Ø
ENTER LOGICAL TAPE NUMBER: 1
NAME: sample
DATE(MM DD YY): 1Ø-14-8Ø
REV NO:      Ø
REEL NO:      1
READY TO RESTORE: nw 1
*** STARTING INDEX ***
ALPHA
BETA
GAMMA
*** END LOGICAL TAPE ***
*** INDEX COMPLETE ***
OK,
```

▶ MAGSAV [options]

MAGSAV writes information from a disk configured to PRIMOS to a 7- or 9-track magnetic tape. When invoked, MAGSAV responds with a series of questions. These questions and appropriate user replies are discussed after the paragraphs that describe the **options** that may be specified with the MAGSAV commands. Optionally, the user may specify: 1024 word or variable length tape records, use of 7-track magnetic tape, keeping track of date-time stamped files and directories, and incremental dumping of a disk to tape. The value of options are:

Option	Meaning
-LONG	Use a 1024 word record size. The default record size is 512 words per record.
-VAR	Allows variable-length records, up to 2048 words; overrides -LONG option. Improves speed of MAGSAV operation. If this option is selected, MAGSAV prints the record size after the REV message in the MAGSAV dialog.

-7TRK Use 7-track magnetic tape format. The default is 9-track magnetic tape.

-UPDT Indicates update; i.e., the DUMPED switch in the UFD entry will be set for files and directories that are saved from disk onto tape. The default is not to set the DUMPED switch.

-INC Indicates incremental dump; that is, only files and directories with the DUMPED switch set to 0 will be saved. The default action is to save all files and directories.

-TTY MAGSAV takes tape unit number from terminal, all other information from current input stream.

MAGSAV requests information in the following order:

TAPE UNIT: The proper response is the physical unit number of the tape (0-7).

ENTER LOGICAL TAPE NUMBER: The response is 1 for the first logical tape, 2 for the second, etc. MAGSAV rewinds the tape, then positions itself correctly. A response of 0 implies the tape is already positioned correctly and MAGSAV takes no action.

TAPE NAME: Any six-character name.

DATE: The response format is **mmbddbyy** where **b** represents a space and **mm** = month, **dd** = day and **yy** = year. The date is checked for validity and rejected if it is not valid. For example, 07 35 03 would be rejected. If not specified, the default is current date. (Carriage return causes system date to be used.)

REV NO: An arbitrary number (usually 0).

NAME OR COMMAND: NAME asks the user what to save. The response is either a treename or one of the alternate action commands. These alternate action commands are:

$A dirname [passwd] [ldisk] [key] Attach to the UFD specified by **dirname** on the disk **ldisk** (use **passwd** if required). To specify a sub-UFD, ATTACH first to its UFD, and then attach to the sub-UFD, giving the **ldisk** number and the key "2". $A does not accept pathnames.

$Q Terminate a logical tape, and return to PRIMOS.

$R Terminate a logical tape, rewind, return to PRIMOS.

$I [filename] [level] Cause an index to be printed. If **filename** is specified, the index is written into the file specified by **filename**. If **level** is specified, then index is to that level. For example: $I 3 prints an index of the MFD, and UFDs and any filenames.

Defaults: if no **level** is specified, then two levels are printed.

If filename is not specified, then the index is printed on the terminal.

$UPDT { **ON** / **OFF** } Save current directory.

ON indicates to set the DUMPED switch of each file MAGSAVed. The default is **OFF**.

$INC	**ON** / **OFF**	If **ON** is specified, MAGSAVe only those files and directories having a set DUMPED switch. Default is **OFF.**
$OLD	**ON** / **OFF**	Causes an "old partition" format tape to be created. (This *should only* be done to send a tape to a pre-Rev. 12 system.) Default is **OFF.**
MFD		Save entire volume.
$VALID	**ON** / **OFF**	Check each name to be sure it conforms to "new partition" file name rules. This should only be done when a MAGSAV is being performed on an "old partition" disk in order that the tape may be restored onto a "new partition" disk.

Example:

```
OK, as mtØ
Device MTØ assigned.
OK, magsav
REV. 17.4
TAPE UNIT (9 TRK): Ø
(Tape not at load point)
ENTER LOGICAL TAPE NUMBER: 1
TAPE NAME: sample
DATE (MM DD YY):
REV NO: Ø
NAME OR COMMAND: alpha
NAME OR COMMAND: beta
NAME OR COMMAND: gamma
NAME OR COMMAND: $r
1 RECOVERED MT IO ERRORS
OK,
```

The MAGSAV rewind command, $R, causes the tape to be rewound and exits to command level.

To save an entire disk, the user must attach to the MFD and respond to the query NAME OR COMMAND: with the name MFD. To save a UFD, the user must attach ($A) to the MFD and give the name of the UFD that is to be saved. To save a file in the UFD the user must attach to the UFD (e.g., $A dirname) and give the name of the file. MAGSAV also saves a disk that contains nested segment directories.

MAGSAV can handle up to 13 levels of directories and subdirectories. Disks or UFDs containing more than 13 levels must be saved in a series of steps.

Except under PRIMOS II, the magnetic tape must first be assigned using the ASSIGN command. Files or directories that are in simultaneous use by other users must not be accessed by MAGSAV, as MAGSAV will ignore such files and so will not write them onto tape.

Error recovery: A separate error message for each recoverable error is not printed. If recoverable errors occur on the magnetic tape when writing tape,the total number of such errors is printed when the end-of-tape is reached.

▶ MAKE *Operator command*

Make formats disks (and partitions of multi-surface disks) for use by PRIMOS. MAKE creates a PRIMOS disk that has the following:

- Disk pack or partition name (packname) as specified by user
- MFD (Master File Directory)
- BOOT (Bootstrap, in Record 0)
- BADSPT (only if badspots are present on disk)
- DOS (which is empty)
- CMDNC0 (which is empty)

See **The System Administrator's Guide** for details.

▶ MAXSCH n *Operator command*

The MAXSCH command is used to set the variable MAXSCH in the system data base named SUPCOM, which controls the circumstances in which the backstop process (part of the scheduler) adds processes to the ready list. The default value of MAXSCH is 3.

▶ MAXUSR n *Operator command*

MAXUSR is a PRIMOS III, or PRIMOS operator command that controls and limits the number of users logged in to the number specified by **n**.

▶ MCLUP *Operator command*

MCLUP invokes the MIDAS Cleanup Utility. For details, see **The MIDAS Reference Guide.**

▶ MDL

MDL punches paper tape of specified sections of memory in a self-loading format that can be read by the Autoload (control panel LOAD) operation (or equivalent operation). MDL tapes load into the same memory locations from which they are punched. Tapes can be punched using locations as low as '34. For details, see Appendix A.

▶ MESSAGE { username / -usernumber } [-NOW]

text of message

The MESSAGE command is used to send or receive messages. Either users or the operator may send messages. Messages may be sent:

- From any user terminal to any user terminal,
- From any user terminal to the supervisor terminal,
- From the supervisor terminal to all users,
- From the supervisor terminal to a specified user,
- From the supervisor terminal to another supervisor terminal on a different node on the network.

User messages

The format of a user-to-user or user-to-operator message is:

MESSAGE { username / -usernumber } [-NOW]

text of message

username is the UFD name a user is logged into. **usernumber** is the number of a specific terminal.

To find out what the usernumbers are for the various terminals issue the STATUS USER command. A list of users, their usernumbers, line numbers, and physical device numbers will be printed.

If you send a message to **username,** all users logged into that name receive the message.

If you send a message to a **usernumber**, only the specific terminal with that number receives the message.

text of message is a single line to be sent. Sending a message produces two lines of information on the receiver's terminal. The top line contains information about the sender; the second contains the text of the message. The format is:

```
***uu hh:mm
   text of message
```

where **uu** is the **username** and **usernumber** and **hh'mm** is the time of day in hours and minutes. For example:

```
***BEECH (55)  11:16
```

If the **-NOW** option is specified, the message is printed immediately on the receiver's terminal.

If the **-NOW** option is not specified, messages are stored in a buffer and printed when the receiver returns to PRIMOS command level.

Setting receive states

Users may set the receive state of their terminal with the MESSAGE command. One of three different states may be selected to control the flow of messages.

MESSAGE -ACCEPT	(enables reception of all messages)
MESSAGE -DEFER	(inhibits immediate messages)
MESSAGE -REJECT	(inhibits all messages)

Setting a receive state to defer or reject messages is useful when you do not want messages to interrupt a terminal session. For example, this can be critical in situations where you are printing the contents of a file. Deferring or rejecting messages in this instance would prevent the message from being printed along with your file's contents.

Sending a message while in MESSAGE -REJECT mode or sending an immediate message while in MESSAGE -DEFER mode is not permitted because the receiver will not be able to respond.

Querying receive states

You may determine what a user's terminal receive state has been set to with the -STATUS option of the MESSAGE command. Issuing the command MESSAGE -STATUS lists users' login names, terminal numbers and receive states.

MESSAGE -STATUS	lists the receive state of all users.
MESSAGE -STATUS username	lists the receive state of all users with the name username.
MESSAGE -STATUS usernumber	lists the receive state of the terminal with the number usernumber.
MESSAGE -STATUS ME	lists the receive state of your own terminal.

Error messages

The following are possible error messages:

BAD MESSAGE

This message may be received if a typing error was made.

UNKNOWN ADDRESSEE

This message may be received if you try to send a message to someone not logged in.

USER NOT RECEIVING NOW

This message may be received if the terminal accept state has been turned off.

USER BUSY

This message may be received if either the terminal buffer or the message buffer is full.

▶ MPACK

MPACK is a MIDAS utility for packing and restructuring MIDAS files. It recovers space occupied by data records and index entries marked for deletion. It also unlocks locked records and restructures the data subfile. For details, see the **MIDAS User's Guide**.

▶ MRGF file-a file-b [file-c...file-e] -OUTF ofile

[-BRIEF
-MINL number
-FORCE
-REPORT pathname]

The MRGF command allows a user to merge ASCII files. Two to five files may be merged. The files may be specified by pathname. One file, **file-a**, is treated as the original file, and it is assumed that changes have been made in this file to produce the other files, **file-b** through **file-e**.

Unchanged lines of text and unconflicting changes between files are copied automatically into the output file, **ofile**. When corresponding lines of text in the specified files differ, the user is asked by the MRGF program to resolve the conflicts. The user's response to these queries determines the manner in which the conflicts arising out of the differences are resolved.

Purpose of MRGF: The MRGF command, along with the CMPF command, helps ease the problems of parallel software development. The MRGF command allows automated merging of program changes, and obviates the need for tedious editing of programs when two (or more) sets of changes made to a program are to be combined. It is anticipated, however, that the resulting merged output will be checked carefully before use.

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 one or more persons.

Parameters	Meaning
file-a	The pathname of the original file. **file-a** is treated as the original file by MRGF (i.e., the file that is the common ancestor of **file-b** through **file-e**).
file-b, file-c, file-d, file-e	Pathnames of files that trace their ancestry to **file-a**.
ofile	The pathname of the merged output file generated by MRGF. It must appear immediately after **-OUTF**.

-BRIEF	An optional control argument that suppresses the printing (or display) at the terminal of differing lines of text within the specified files. (Only the file identification letters and line numbers are printed.)
-MINL number	Sets the minimum **number** of lines that must match, after a difference in the files being merged, in order to resynchronize all file merging. The default value is -MINL 3.
-FORCE	Causes file-b to be the preferred file if conflicts exist between several files. When -FORCE is used, the user is never asked by MRGF to resolve a conflict. (See the following paragraphs for a discussion of resolving conflicts.)
-OUTF	Immediately precedes **ofile**, the output file. **-OUTF** is mandatory.
-REPORT pathname	Produces a file, named **pathname**, that contains the differences (in lines of text) between files encountered while merging. Resolvable differences are not displayed or printed at the terminal. User-resolvable differences (conflicts) are written into the report file, **pathname**, and they are also displayed or printed at the terminal.

MRGF usage: **file-a** is treated as an original file (i.e., as a file that is the common ancestor of **file-b** through **file-e**). **file-a** is compared line by line with each of the other files. Lines that match in all files are copied into **ofile**. When differences are found between specified files, MRGF attempts to get all files back in synchronization. Rematching is completed only when a certain minimum number of lines match in all files. This minimum number may be set with the -MINL control argument.

After resynchronization is complete, selection of lines to be output must take place. If only one file differed from **file-a**, the changes in that file are copied into **ofile**. If all files differed identically from the original, those changes are also copied. If conflicting changes are found in several files, (or if only one file is being merged with the original), the user can select manually the lines that are to be copied into **ofile**. If the -FORCE control argument is used, the user is not queried; and the changes in **file-b** are considered the preferred changes to be inserted into **ofile**

If the -FORCE control argument is not used, the differing lines from each of the files are reported. Each line from **file-a** is identified by preceding it with the letter A and the line number of that line. Lines of **file-b** through **file-e** are similarly identified, using the letters B through E, respectively. The -BRIEF control argument causes only the identification letter and the line numbers of the differing lines to be printed. After an unresolvable discrepancy is reported, EDIT mode is entered to allow the user to select lines to be placed in **ofile** (refer to the following paragraphs). After selection (either automatic or manual) is completed, the line-by-line comparison continues.

If the -REPORT control argument is used, the resultant report file contains all discrepancies between files (i.e., both the resolvable and the unresolvable differences). Unresolvable differences are always displayed on the user's terminal as well. Resolvable differences, however, are never displayed on the user's terminal. The action taken by MRGF (or the user) is placed in the report file following each discrepancy.

Manual selection of changes: After each unresolvable difference is displayed, EDIT mode is entered. The user must select which lines are to be inserted into **ofile** by issuing the following commands:

A	Insert all of the differing lines in **file-a.**
B	Insert all of the differing lines in **file-b.**
C	Insert all of the differing lines in **file-c.**
D	Insert all of the differing lines in **file-d.**

E	Insert all of the differing lines in **file-e**
An	Insert line **n** of **file-a**
Bn	Insert line **n** of **file-b** (similarly for **file-c** through **file-e**).
Am,n	Insert lines **m** through **n** of **file-a** (similarly for **file-b** through **file-e**).
PA	Print all of the differing lines in **file-a** (similarly for **file-b** through **file-e**).
PAn	Print line **n** of **file-a** (similarly for **file-b** through **file-e**).
PAm, n	Print lines **m** through **n** of **file-a** (similarly for **file-b** through **file-e**).
OOPS	Undo all previous editing for this discrepancy.
GO	Terminate editing and proceed with merge.
QUIT	Terminate editing, close all files, and exit from MRGF.

In addition to the above commands, new text can be inserted at any point in a difference resolution by entering a blank line. INPUT mode is entered, and lines typed are copied into **ofile**. A blank line will terminate input. No text editing can be performed on lines that are copied or entered in INPUT mode. No tab character expansion is performed on lines entered in INPUT mode.

Line length: The MRGF command operates on compressed lines of any length. It assumes that files of common ancestry contain lines compressed in identical fashion. It is, however, possible for a mismatch to occur between two lines that appear identical, but were compressed differently.

For example, consider the following three files:

```
FILEA       FILEB          FILEC

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
```

A MRGF of these files would produce the following:

```
OK, MRGF FILEA FILEB FILEC -OUTF FILEX

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.
```

In the above example, rust colored lines were typed by the user. The merged output file from the above MRGF would appear as follows:

```
The
quick
red
fox
jumps
over
the
sleeping
dog
```

If the -FORCE control argument had been used in the example given, the same merged output would have been produced. However, the change from FILEB would have been inserted automatically, and the user would not have been queried.

▶ NCOBOL

NCOBOL is a COBOL compiler that generates R-mode code and uses the non-shared COBOL library, NCOBLB. Its options and defaults are the same as those of the V-mode COBOL compiler, COBOL.

▶ NET *Operator command*

The operator uses the NET command to activate, assign, and deactivate half duplex (HDX) network connections. For details, see **The PRIMENET Guide**.

▶ NETCFG *Operator command*

The System Administrator uses the NETCFG command to establish the configuration of a PRIMENET network. For details, see **The PRIMENET Guide**.

▶ NETLINK

You may connect to any system on the Public Data Network by using the NETLINK command. This means that systems other than Prime systems and software other than PRIMENET software may be accessed across geographic boundaries. Other sites or other networks as well as jobs within these other sites and networks may be accessed.

Several NETLINK commands exist to aid users in using other systems and networks. There are basic commands and advanced commands. Basic commands allow you to enter and exit the remote systems. Advanced commands allow you to:

- Transfer files across networks,
- Set data transmission characteristics,
- Print the status of your connection,
- Connect to and use up to four different remote systems at the same time,
- Specify the various fields of the connect packet when data transmission characteristics of a foreign system differ from that of Prime's.

Only NETLINK's basic usage will be presented here. For a list of all commands and error messages see **The PRIMENET Guide**.

NETLINK usage

The basic steps to using NETLINK are as follows:

1. Enter NETLINK Command mode by issuing the NETLINK command. When

Command mode is entered, the @ prompt appears.

2. Connect to the remote system by issuing the **NC address** or **C address** command. **NC** means no reverse charge. This is required for many international calls. **address** is either the host address assigned by the Public Data Network or a PRIMENET system name. For example 'NODE 1' and '617 74' are both valid addresses.

   International calls must be specified as follows: 'county identifier: host address.' For example, '2080:12300011'

   When a connection has been established, the message: **address Connected** appears.

3. Login to the system as you would normally, entering any validation codes as required.

4. Once you finish a terminal session, logout as you would normally. The message: **address Disconnected** appears. When a connection to a remote host has been terminated by logging out, Command mode is re-entered and the @ appears. You may now connect to another site or return to PRIMOS.

5. To return to PRIMOS enter the **QUIT** command.

NETLINK example

Below is an example of a basic terminal session.

```
OK, netlink
[NETLINK  Rev 18.1]

@ c node1
Circuit #1
NODE1 Connected

PRIMENET 18.1 NODE1
login beech secret

PRIMOS Version 18.1
BEECH  (31) LOGGED IN AT  14:43 11Ø48Ø
Enter validation code: agent
   .
   .
   .    /* continue with normal terminal session
   .
   .


OK, logout
BEECH  (31) LOGGED OUT AT  14:44 11Ø48Ø
TIME USED=  Ø:Ø1   Ø:Ø1   Ø:Ø3

WAIT...
NODE1 Disconnected


@ quit
OK,
```

NETPRT

The functionality of the NETPRT command has been subsumed into the LOGPRT command. Use the LOGPRT command to read or print the network event log, NETREC. For details, see the **System Administrator's Guide.**

NSED

NSED is the non-shared version of the text editor. It is identical in function to ED, the shared Editor. PRIMOS II, which does not support the shared Editor, can use NSED.

NUMBER

NUMBER numbers or renumbers statements in a BASIC program. NUMBER asks the user:

```
INTREENAME, OUTREENAME, START, INCR,
```

The user gives four responses on one line:

intreename	The pathname of the input file (i.e., the file that contains the BASIC program with the statements to be renumbered).
outtreename	The name of the output file. If **outtreename** is omitted, output is to the input file.
start	The starting statement number, from 1 to 9999. If **start** is omitted, the value 1 is assumed.
incr	The statement number increment, from 1 to 9999. If **incr** is omitted, the value 1 is assumed.

If **incr** is specified, **start** must be specified also.

The maximum line length that NUMBER can handle is 75 characters, plus 5 for the line number. Lines longer than 75 characters are truncated. Only those BASIC programs with their commands in UPPERCASE characters are renumbered correctly.

Examples:

Assume an input file contains the following statements:

```
11 PRINT 'AB'
12 INPUT A
30 PRINT 'H'
35 INPUT H
40 B=H/A
50 PRINT ' ',B
55 IF b<>0 THEN 11
99 END
```

Then, the following sequence of commands:

```
OK, NUMBER
INTREENAME, OUTTREENAME, START, INCR,
FOO FOOBAR 10 5
OK,
```

produces the following output:

```
1Ø PRINT 'AB'
15 INPUT A
2Ø PRINT 'H'
25 INPUT H
3Ø B=H/A
35 PRINT ' ',B
4Ø IF b<>Ø THEN 1Ø
45 END
```

The input file may be only partially numbered. In such a file, statements are numbered in the order of their occurrence.

Error messages: The following messages may be printed, if an error occurs:

Messsage	Remarks
BAD PARAMETERS	If either **start** of **incr** are specified with more than 4 digits, NUMBER requests a new parameter line.
BAD SYNTAX	If the value of either **start** or **incr** is invalid, NUMBER requests a new parameter line.
XXXXXX NOT FOUND	The specified input file does not exist. Control returns to PRIMOS.
XXXX DUP LINE NUMBER	XXXX occurs as a line number more than once. Control returns to PRIMOS.
INPUT FILE NULL	The specified input file is empty. Control returns to PRIMOS.
MEMORY OVERFLOW	There is not enough memory to contain a map of line numbers. Control returns to PRIMOS.
LINE NUMBER OVERFLOW	A new line number greater than 9999. Control returns to PRIMOS.

▶ OPEN [pathname] funit key

OPEN opens the file unit specified by **funit** (between 1 and 177 octal), associates it with the specified **pathname**, and assigns a status according to the **key**.

The **key** parameters consist of octal values for the type of file and the action to be taken when the file is opened. The format of **key** is as follows:

New file (file type) key; octal values are:

0000	File will be sequential file (SAM)
2000	File will be direct access file (DAM)
4000	File will be a SAM segment directory
6000	File will be a DAM segment directory
10000	File will be a UFD (avoid this; use CREATE command instead)

Action Key; octal values are:

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 (no pathname)

Reference key; values are:

0	File is entry in current UFD
100	File is entry in segment directory open on **funit**
1000	Change open mode of **funit**

The octal values for file type, action, and reference are logically ORed to form key. For example, the key for opening a DAM file for writing, in the current UFD, is formed by ORing together the values:

2000	Direct Access file (DAM)
0000	In current UFD
0002	Open for writing
2002	(The resulting key)

See also: **CLOSE.**

▶ OPRPRI {1 / 0}

Operator command

OPRPRI (operator privilege) allows certain commands to be issued at the supervisor terminal. Under PRIMOS, OPRPRI 1 must precede the SHARE or LOOK commands. The command line: OPRPRI 0, resets the protection against these commands.

▶ OWLDSC [-FAST] [-NOLOCK] [-REPORT]

The OWLDSC command 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.**

▶ PASCAL pathname [options]

The PASCAL command loads the PRIME PASCAL compiler and compiles the object program from an ASCII source file named **pathname**. For complete details about programming language PASCAL and compiler **options** see **The PASCAL Reference Guide.**

▶ PASSWD [owner-password] [nonowner-password]

The PASSWD command replaces any existing passwords in the current directory with two new passwords. The first is the **owner-password** the second is the **nonowner-password**. The **nonowner-password** is optional. If it is not specified, the **nonowner-password** becomes blanks. The PASSWD command must be given by the owner while attached to the directory. A nonowner cannot give this command. Passwords may be of any length; but they are matched by the first six characters only.

Passwords may be specified in lower or uppercase. To specify a lowercase password, type the password in lowercase and enclose the password in single quotes. To specify an uppercase password, type the password in uppercase characters; do not enclose the password in single quotes.

Under PRIMOS II, only the owner password may be given.

Example:

```
OK, A GOUDY OLDPW
OK, PASSWD US THEM
OK,
```

CAUTION

The MAGRST command cannot restore directories with passwords unknown to the user of MAGRST.

▶ PHANTOM command-file [funit]

A user may initiate a phantom user to perform a job. A phantom user is similar to any other PRIMOS user, but has no terminal associated with it; all controlling input is read from a **command-file** instead of a user terminal. The **command-file** may be a CPL program or a command input file. It may be specified as a pathname or as a file in the user's current directory. The file specifies the sequence of commands and/or user program invocations and necessary input data specifications to complete a given job.

The last command in **command-file** must be LOGOUT (not COMINPUT -TTY or COMINPUT -END). Otherwise, the phantom will report an abnormal ending when it finishes processing the file.

funit may be specified for command input files, to control the file unit on which **command-file** is opened. It may not be specified for CPL programs.

The phantom user feature is useful for running programs that are not interactive, and therefore do not require the services of a terminal. However, terminal output should be directed to a COMOUTPUT file.

When PRIMOS is started up, a fixed number of phantom users is specified. The line printer spooler, or the card reader spooler, is usually run as a phantom user, thereby releasing a terminal for interactive work.

Startup of PHANTOM The PHANTOM command checks if there is a process available for a phantom user to be logged in. If no free processes are available, the message:

```
No phantoms are available.  FILENAME
```

is printed at the user terminal. If a process is available, the phantom user is logged into the login directory of the user who invoked the phantom. Then, the phantom feature of the operating system attaches to the user's current directory. If the **command-file** specified in the phantom command is a command input file, it is opened on File Unit 6 (or on the specified **funit**). If the command-file is a CPL file, it is opened on some available unit. PRIMOS takes all further commands from the file specified by **command-file** in accordance with COMINPUT or CPL operation. An example of phantom startup is:

```
OK, ph test.phant
PHANTOM is user 1Ø6
OK,
```

It is suggested that the COMOUTPUT command be invoked within the phantom-command-file to keep a historical record of the phantom's processing. A process running as a phantom user *must not* perform any terminal input. An attempt to read input from a terminal causes the command file to abort and causes the phantom user to be logged out. If this condition occurs, the logout message at the system terminal is preceded by the line:

```
PHANTOM TTY REQUEST
```

An error that causes the command file to abort also causes the phantom user to be logged out.

Any terminal output that is generated by the phantom user program or directed to the user terminal by system commands, such as LISTF, is ignored, unless the COMO command is invoked in the command file.

A user may monitor the status of any phantom user by the STATUS command. For each phantom user that is logged into the same login UFD, STATUS prints the UFD name followed by the user number of that phantom in decimal. When the user phantom job is complete, it is logged out, and will no longer appear in the output printed by the STATUS command. If a user wishes to stop a phantom user that was started at his terminal, the command:

LOGOUT -uu

must be issued, where **uu** is the number of the phantom user as reported by the PHANTOM command when it was invoked.

A user may log out, return later, and log in to the same UFD. The STATUS and LOGOUT commands may be used as before to control the phantom.

Any phantom or user may be logged out by use of the LOGOUT command at the supervisor terminal.

The PHANTOM command may be issued from a CPL /program or a command file. Command files running in phantoms may also include PHANTOM commands (i.e., phantom command files may be chained in a manner similar to COMINPUT command files).

When a phantom logs out, it attempts to send a notification of its logout to the user who started it. A sample message, following an orderly logout, might be:

```
PHANTOM 106 NORMAL LOGOUT AT 11:34
time used= 0:3  0:0  0:0
```

The phantom can send this message to the user's terminal only if the user is 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 LO$CN and LO$R. (For information on these subroutines see **The PRIMOS Subroutines Reference Guide.**)

See also: JOB

▶ PHYRST *Operator Command*

The PHYRST command copies partitions to disk as saved by PHYSAV on magnetic tape.

For a complete description of this command see **The System Administrator's Guide.**

▶ PHYSAV *Operator Command*

The PHYSAV command copies the contents of one or more assigned disk partitions to magnetic tape.

For a complete description of this command see **The System Administrator's Guide.**

▶ PL1G pathname [options]

PL1G is Prime's compiler for subset G of PL/I. It accepts either a filename or a **pathname** However neither the **pathname** nor any filename created from it may contain more than 32 characters.

At Rev. 17.2 PL1G generates V-mode and I-mode code. Thus, SEG must be used to load the compiled program. Moreover, a new library (PL1GLB) must be loaded before loading the regular library. The commands are:

```
$LI PL1GLB
$LI
```

As distributed, PL1G uses the following default options: -B YES -L NO -64V -OPTIMIZE -UPCASE. (The defaults may be changed by using the program TOOLS>PL1GDF.)

For a listing of all options and their uses, see **The PL1G Reference Guide.**

▶ PM

The PM (Post Mortem) command is used to print or display the contents of the RVEC vector (described in Appendix A). PRIMOS first prints labels for the items in RVEC, then prints the values on the line in the same order.

For the Prime 350 and up, 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.

Example:

```
OK, PM
SA,EA,P,A,B,X,K=
100 5521 6333 0 0 0 34100

PB,SB,LB,XB:

2000(3)/6333  4000(3)/112000  4000(0)/2214  0(0)/0
OK,
```

The above example of PM under PRIMOS shows a PB of 4000(3)/1106, which indicates: ring 3, segment '4000 octal. The word number portion of PB indicates the same number as the P parameter of PM. This number, which is the same as the P parameter, specifies the location within the segment to execute the next instruction upon possible receipt of a START command.

Note

PM will not give an accurate picture of the machine state of a program if the program has been halted by an on-unit that does not allow the Static mode overseer to update the PM data. This would occur if a user on-unit returned to command level by calling COMLV$. However, the DMSTK command will always produce an accurate display of the program's state.

▶ PMA pathname [options]

PMA loads the Prime Macro Assembler and starts assembly of a source file specified by **pathname**

Option	Meaning or Remarks
-INPUT filename-1	Specifies the name of the input file. The default is the pathname following the command PMA.
-LISTING filename-2	The name of the listing file. The default is filename-1 . list or L_filename-1. Use -LISTING NO for no listing.
-BINARY filename-3	The name of object file. The default is filename-1 . bin or B_filename-1. Use -BINARY NO for no object file.
-EXPLIST	Generates full assembly listing. (See the **Assembly Language Programmers Guide** for further details.)
-ERRLIST	Generates errors-only listing.
-RESET	Resets all A-, B-, and X-Register settings.
-XREFL	Generates complete concordance.
-XREFS	Generates partial concordance; only symbols actually referenced are listed.

For a complete discussion of the assembler, including register settings, see the **Assembly Language Programmer's Guide.**

▶ POWER

Invokes the PRIME POWER data management facility. For information, see the **PRIME/POWER Guide.**

▶ PRERR

PRERR prints the message stored in ERRVEC and the first six locations of ERRVEC in octal. The PRERR command is useful in debugging a program. 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 PRIMOS Subroutines Reference Guide** , for a description of ERRVEC.

Using the system subroutine ERRSET (see **The PRIMOS Subroutines Reference Guide**), a user may set the content of the error message and have the message printed or not printed, depending on the alternate return being zero or nonzero, in a user subroutine. If the user routine was the last routine to set ERRVEC, PRERR prints the user-stored message.

Example:

```
OK, prerr
45 1 Ø Ø 142722 120305

OK,
```

Note

The PRERR command will not display useful information following such conditions as Access Violation faults or Illegal Segment Number faults. In these cases, the needed information will be printed as part of the system diagnostic for the error condition. It can also be obtained by using the DMSTK command.

▶ PRMPC pathname

PRMPC causes the file specified by **pathname** to be printed on a MPC parallel interface printer configured to PRIMOS. The printer (PR0) must be ASSIGNed before the PRMPC command is invoked.

▶ PROP

The PROP command allows the operator to control the spooler phantoms. It also allows users to discover the names and environments of these phantoms: that is, it allows them to find out how their system's printers and/or plotters behave.

Users can give two forms of the PROP command:

PROP -STATUS

which lists the spooler phantoms and tells which ones are running; and

PROP printer-name -DISPLAY

which provides information on a phantom's environment. For example:

```
OK, prop example -display
[PROP rev 18.Ø]

DEVICE: PRØ
PAPER:  3-PLY
FORM:
    'ORDERS'
```

```
DEST:

   RECEIVING
   OFFICE-300
MESSAGE:
This is an example environment.

COMOUT: OFF
UPCASE: ON
PRINT:  ON
PLOT:   OFF
LENGTH:     53
LARGE:      40
LIMIT:    5000
UPPER:      70
LOWER:       0
HEADER:      2
WIDTH:      88
LINES:      66
OK,
```

Within the PROP display, the parameters of interest to most users are:

Parameter	Definition
DEVICE:	Name of printer phantom. (Can also be used in a spool "-AT" option.)
PAPER:	Type of paper mounted on printer. (Blank signifies default type.)
FORM:	Specifies synonyms for PAPER type that users can use with SPOOL's -FORM option. In this case, ORDERS is an acceptable synonym for 3-PLY. This parameter does not always appear.
DEST:	Synonyms for printer-name that users can specify in the -AT option of the SPOOL command. (The synonyms tell the operator where to deliver the printout.) This parameter is optional; it appears only if the sytem administrator has established a list of synonyms.
UPCASE:	If on, printer prints all characters as uppercase. If off, prints both upper and lowercase.
PRINT:	If on, phantom controls a printer.
PLOT:	If on, phantom controls a plotter.
LARGE:	Files of less than this number of records receive priority in the spool queue.
LIMIT:	No files containing more than this number of records will be printed by this phantom.

Other Parameters are:

LENGTH:	Specifies number of lines printed per page.
WIDTH:	Specifies number of columns per page.
HEADER:	Specifies number of header pages to be printed.
MESSAGE:	This message is printed on each header page.
COMOUT:	If on, the phantom is keeping a COMOUTPUT file of its activities.

UPPER: Specifies highest logical disk on which phantom will look for its SPOOLQ UFD.

LOWER: Specifies logical disk number at which phantom will start search for SPOOLQ UFD.

LINES: Actual number of lines per page. (Zero becomes default length, which is "LENGTH" plus 13 lines.)

For a discussion of PROP commands available to the operator, see **The System Administrator's Guide.**

▶ PROTEC pathname key1 key2

Users (hereafter called owners) have the ability to open their files and directories to other users, giving controlled access rights to their files. This declaration of access rights can be made on a per-file basis. Access rights to a file are declared and specified through the PASSWD and PROTEC commands.

pathname The name of the file to be protected
key1 An integer that specifies the owner's access rights to **pathname**
key2 An integer that specifies the nonowner's access rights to **pathname**

The values and meanings for **key1** and **key2** are:

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

Example:

```
OK, PROTEC MYPROG 7 1
OK, PROTEC OLDIS 7 7
```

gives the owner all access rights of MYPROG, nonowners read-only access rights to MYPROG, and gives both owners and nonowners all access rights to the file OLDIS.

Note

The default protection keys associated with any newly created file or UFD are: 7 0 (owner is given all rights and nonowner is given none). However, if PROTEC is given without arguments, values are set to 0 0 (neither owner nor non-owner has any rights).

The following example is intended to give a user an idea of the use of the PASSWD and PROTEC commands:

```
OK, ATTACH JHNDOE
OK, PASSWD US THEM
```

Gives owner password US and non-owner password THEM to current UFD.

```
OK, PROTEC TIMING 7 0
```

Gives JHNDOE all access, non-owners no access to TIMING.

```
OK, PROTEC MYPROG 7 1
```

Gives owner access=all, nonowners access=read.

```
OK, PROTEC OLD 7 7

OK, LOGIN MSMYTH

MSMYTH (2) LOGGED IN AT 11:34 1Ø1Ø79

OK, ATTACH JHNDOE THEM

OK, LISTF

UFD = <MISCEL>JHNDOE 2 NONOWN

TIMING   FUNCT  MEMOS  MYPROG  OLD

OK, DELETE TIMING
```

nonowners access=all, owner access=all.

```
Insufficient access rights. TIMING
```

MSMYTH, a nonowner, cannot even read timing since she has no access.

```
ER! ED MYPROG
p2;
.NULL
C PROGRAM TO TEST DATA
C JOHN DOE Ø2 Ø2 Ø7
INPUT

C WITH CHANGES INSERTED
C BY MSMYTH

EDIT
```

MSMYTH can enter editor and read MYPROG since read access has been granted.

MSMYTH attempts to change MYPROG; he seems to have succeeded.

```
FILE MYPROG
Insufficient access rights. MYPROG (OPENR)
?
Q
FILE MODIFIED, OK TO QUIT? yes
OK, ED OLD
EDIT
;
INPUT
```

Cannot change file because write access is denied by JHNDOE. Might as well quit.

```
C CHANGES BY MSMYTH WILL BE RECORDED HERE
;
EDIT
FILE OLD
OLD
OK,
```

As all access has been granted for OLD, changes are made successfully, however, MSMYTH cannot create a new file (i.e., it is not possible to use editors file command with a file name other than OLD).

▶ PRSER pathname

PRSER causes the file specified by **pathname** to be printed on the serial interface printer configured to PRIMOS. The printer (CENPR) must be ASSIGNed before the PRSER command is invoked.

▶ PRTDSC station-1 station-2...

The PRTDSC command invokes the printer emulation program on systems where DPTX/DSC is running. Printer output is spooled with form type equal to to the first six characters of the station name. For details, see the **Distributed Processing Terminal Executive Guide.**

▶ PRVER pathname

PRVER prints a file specified by **pathname** on a printer/plotter configured to PRIMOS. The plotter must be assigned before the PRVER command can be issued.

Example:

```
OK, ASSIGN PLOT
OK, PRVER SALES.GRAPH
```

assigns the printer/plotter and prints the file at the printer/plotter.

▶ PSD

PSD loads and starts Prime Symbolic Debugger, an interactive debugging program that assumes control and waits for a command string. For details, see the **Assembly Language Programmer's Guide**. Commands of Prime's debugging utilities (PSD, TAP, and VPSD) are summarized in Appendix C.

To return to PRIMOS, type Q and Carriage Return at the user terminal.

▶ PSD20

PSD20 is a version of PSD for 16K PRIMOS II. It is identical to PSD but occupies locations '17760 to '26552. It is described in **The Assembly Language Programmer's Guide**.

▶ PTCPY

PTCPY is a utility program that duplicates and verifies paper tapes using the high-speed reader-punch. Operation is controlled by P-register and sense switch settings. PTCPY is an external command. For details, see Appendix B.

Under PRIMOS, the command ASSIGN PTR and ASSIGN PUN must be given before PTCPY is invoked.

▶ PT45DSC

The PT45DSC command invokes the PT45 interface program, which allows a PT45 terminal to emulate an IBM3277 Model 2 display station on systems where DPTX/DSC is running. For details see the Distributed Processing Terminal Executive PTU.

▶ RDY [option 1] [option 2]

The RDY command allows users to choose the prompt messages they want displayed at their terminals and in their COMOUTPUT files.

The default message (the brief form) consists of the message OK, or ER! The long form includes clock time, the amount of CPU time used since the last prompt, and the amount of I/O time used since the last prompt.

If the user is at a command level above 1, the level number is also printed out. (This happens if the user has interrupted or terminated programs by typing CONTROL-P, or if error-handling mechanisms such as the condition mechanism have interrupted a program.) If the level is marked as static mode, a plus sign (+) follows the number. (This means that the last command executed was an external command.) Prompt messages may also be suppressed, so that they do not print at all.

The RDY command is given with one or more **options** to change the settings governing the printing of messages. It may also be given without **options** to produce a single long-form prompt.

Option	Function
-LONG	Switches to the long format of prompt message.
-BRIEF	Switches to the short format. This is the default at login.
-OFF	Suppresses prompt messages.
-ON	Re-enables the printing of prompt messages. Unless the -LONG or -BRIEF option is given with the -ON option, messages will appear in the format last specified.
-READY_LONG xxx	Changes the text portion of the long ready message to **xxx**. Default at login time is "OK."
-READY_BRIEF xxx	Changes the text portion of the brief ready message to **xxx**. Default at login time is "OK."
-ERROR_LONG xxx	Changes the text portion of the long error message to **xxx**. Default at login time is "ER".
-ERROR_BRIEF xxx	Changes the text portion of the brief error message to **xxx**. Default at login time is "ER!".

The new prompt message, **xxx**, may be up to 20 characters in length. If it contains special characters or embedded blanks, it must be enclosed in single quotes. For example:

```
RDY -RB Satisfactory. -EB Pfui!
Satisfactory. XYZZY
Not found. XYZZY (std$cp)
Pfui!
```

▶ REMOTE *Operator command*

The operator uses the REMOTE command from the supervisor terminal to permit or deny remote (network) access to local file system partitions (disk volumes). See **The System Administrator's Guide.**

▶ REN

This internal command is used to re-enter a subsystem following a QUIT or an error condition. It takes no arguments.

For REN to succeed, the subsystem being re-entered must have defined an on-unit for the condition REENTER$ that can GO TO the appropriate point within the subsystem. At Rev. 18, few subsystems have defined such on-units. If no on-unit exists, the REN command fails, and the user is returned to PRIMOS command level.

REPLY

Operator command

The operator uses the REPLY command to send messages to users in response to magnetic tape assignments requests. For details, see **The System Administrator's Guide.**

RESTOR pathname

The RESTOR command restores a runfile **pathname** from disk to memory, using the RVEC parameters SAVEd with the file.

Note

Do not use RESTOR to restore a 64V segmented mode runfile, use the RESTOR subcommand of SEG.

Example:

```
OK, rest *bench9
OK, pm
SA,EA,P,A,B,X,K=
100 25445 1425 0 0 0 14100

PB,SB,LB,XB:
4000(3)/1425  4000(3)/30012  4000(0)/1040  0(0)/0
OK,
```

RESUME pathname [arguments...]

RESUME runs (executes) the program specified by **pathname**. The program may either be a CPL program (in which case the name of the program must end in .CPL) or a runfile. If **pathname** does not end in .CPL, RESUME first appends .CPL to **pathname** and searches for that file. If RESUME finds the file, it executes it as a CPL program. If RESUME does not find the file, it executes **pathname** as a runfile.

Arguments are defined by the program being executed. If the file is a runfile (not a CPL program) the form of the arguments is:

[p] [a] [b] [x] [keys] [program arguments...]

where [p] [a] [b] [x] and [keys] may be used to set new values for the RVEC. (See Appendix A for details.) The program's runfile is loaded from disk to memory, using the SAVEd values of SA and EA. RVEC is loaded from the SAVEd RVEC parameters or from any new values specified in the command string. The processor registers and keys are then set from RVEC and the program is started at location **p**. Non-numeric arguments are passed to the RESUMEd program.

Note

Do not use RESUME to resume a 64V segmented mode program; use SEG instead. For further details, see SEG in the section.

If the file is a CPL program, **arguments** will not be interpreted as RVEC settings. Instead, they will be passed to the program as CPL arguments when the program executes. (For more information on CPL programs, see the **CPL User's Guide**.)

RJ1004
RJ200UT
RJ7020
RJX80
RJGRTS
RJHASP

Each of these commands is a send utility command for the Remote Job Entry system. Send utility commands are used to submit jobs to remote computer sites.

Each command is for a separate Remote Job Entry emulator. The RJE systems emulated by Prime are:

RJ1004	Univac 1004
RJ200UT	CDC 200 UT
RJ7020	ICL 7020
RJX80	IBM 2780 and 3780
RJGRTS	Honeywell GRTS
RJHASP	IBM HASP

For the details of all these RJE emulators, see the **Remote Job Entry Guide.**

▶ RLS [option]

The RLS command is used to discard unwanted stack history. (The stack history is a record of the calls and returns created by user commands. It is automatically saved by PRIMOS.) **option** determines how much of the stack is to be released, as follows:

-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, and must be less than the current level number.
-LEVELS n	Releases **n** levels. The new stack level will be the current level minus **n**. Since this must be a positive decimal integer, **n** must be a positive decimal integer such that the current level is -n≥1. Default is -LEVELS 1.

If no options are given, one of two things happens. If the user's most recent command was an internal command, then the current level of the stack is released. If the user's most recent command was an external command, then the history of that command is released, but the level number of the stack is not changed.

If a large number of interrupts occurs, the stack grows large and unwieldy. If this happens, PRIMOS warns you that you are NOW AT COMMAND LEVEL nn. TO RELEASE USE RLS. (LISTEN _). If the stack continues to grow and overflows its segment, PRIMOS re-initializes it automatically, sending the message "User environment re-initialized. (FATAL$)." Since most users don't need the stack history preserved, it is usually better to re-initialize the stack yourself when it begins to grow large than to wait for PRIMOS to do it for you.

▶ RPG

RPG invokes the Prime RPGII compiler. Refer to the **RPGII Programmer's Guide.**

▶ RSTERM [-INPUT] [-OUTPUT]

The RSTERM command empties the user terminal's read (input) and/or write (output) buffer. That is, it may be used to throw away typed-ahead input or output pending terminal characters.

Specifying **-INPUT** empties the input buffer. Specifying **-OUTPUT** empties the output buffer. Specifying neither option empties both buffers.

The command is useful in CPL programs for reinitializing the terminal state when a condition handler for the QUIT$ condition is specified.

▶ RUNOFF [pathname]

RUNOFF, Prime's text formatter, accepts commands to control margins, indentation, line spacing, column width, page numbering, running heads, and many other features of an ASCII source file. The commands may be entered from the terminal at run time or be edited into the source file. RUNOFF produces a formatted output to the terminal, or to a designated disk file. For detailed information, refer to the **New User's Guide to EDITOR and RUNOFF.**

▶ SAVE pathname [sa] [ea] [pc] [a] [b] [x] [keys]

The SAVE command saves the content of memory from **sa** (starting address) to **ea** (ending address) as a file named **pathname**. ("Memory" is actually segment '4000.) The other values are the RVEC parameters described in Appendix A. If any parameters are not specified, the existing values of RVEC are taken from the current register set and are stored with the program. The RVEC parameters are used to initialize the processor registers and keys when the program is RESTOREd or RESUMEd.

Do not use SAVE to save 64V segmented mode runfiles. Use the SAVE subcommand of SEG instead. Refer to the **LOAD and SEG Reference Guide.**

Note

All FORTRAN programs begin with ELM (Enter Load Mode). If macro assembler (PMA) user have ELM as the first instruction in the program, there is no need to set the keys after loading. The preferred way to save a memory image is to use the loader SAVE command.

▶ SCHDEC [[-SCHEMA] schema-name [-LISTING] output-file]

Invokes the DBMS Schema Decompiler. For information, see the **DBMS Schema Reference Guide.**

▶ SCHED pathname

Invokes the schema editor (SCHED), an interactive processor that allows a database administrator to alter the definition of a database. For information, see the **DBMS Administrator's Guide.**

▶ SCHEMA source-filename [-VOL volume-name]

Invokes the DBMS Schema DDL compiler. For information, see the **DBMS Schema Reference Guide.**

▶ SEG [pathname]

SEG invokes a utility for loading, modifying, running, and sharing segmented (V-mode and I-mode) programs. SEG **pathname** executes the V-mode runfile specified by **pathname**.

PRIMOS assigns segments to a user as they are accessed. These are not reassigned until either the LOGOUT or DELSEG command is issued.

The maximum number of segments available to a user program is between 32 and 256. It is a configuration dependent parameter, and is available from your System Administrator. User segments start at segment 2048 (4000 octal). Segments in other ranges are for PRIMOS itself, for shared code, and reserved for expansion. Information on installation of shared code is in **The System Administrator's Guide.**

A complete discussion of SEG is given in the **LOAD and SEG Reference Guide.**

▶ SETIME -mmddyy -hhmm *Operator command*

SETIME is an operator command that sets the system date and time of day.

▶ SETMOD { -USER / -OPERATOR / -NOASSIGN } *Operator command*

The SETMOD command determines whether users can assign their own tape drives or not. If SETMOD is set to **USER** (the default), users can assign tape drives from their terminals by the ASSIGN command. If SETMOD is set to **OPERATOR**, the use of the ASSIGN command to

request a magnetic tape assignment prints the request 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. If SETMOD is set to **NOASSIGN**, no magnetic tape assignments are possible. In this mode, use of the ASSIGN command produces a message at the user's terminal saying that tape drives cannot be assigned.

▶ SET_VAR name (:=) value

The SET_command defines a variable and places it and its value in the global variable file. Global variables are the only variables you can use at command level.

name is any legal variable name, up to 32 characters long. Names of global variables must begin with a dot (.).

Value can be:

- a character string. At command level, the string must be short enough so that the entire command line does not exceed 160 characters. In CPL programs, the string may be up to 1024 characters long. A string must be enclosed in single quotes if it contains blanks or special characters. The single quotes are included in the character count.
- a numeric character string representing an integer between the values of (-2**31) + 1 to 2**31 + 1.
- a character string consisting of the logical value TRUE or FALSE.

The assignment symbol (:=) is optional. The command defines the variable **name** if it was undefined, and assigns it the value **value**.

For example:

```
OK, set_var .a alpha
OK,
```

defines the global variable .A and assigns it the value ALPHA.

You can use the SET_VAR command interactively, at command level, to define global variables. Or, you may use it inside a CPL program to define either global or local variables.

See also: **DEFINE_GVAR, DELETE_VAR, LIST_VAR**

▶ SHARE [pathname] segno [access-rights]

Operator command

The PRIMOS SHARE command is used for incorporating files into shared segments. It is issued only from the supervisor terminal. The command OPRPRI 1 must be given for SHARE to work.

The process of incorporating shared code into PRIMOS involves the use of both SEG's 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. Then those R-mode runfiles below segment '4000 must be incorporated into PRIMOS using the PRIMOS SHARE command. For complete details, refer to **The System Administrator's Guide**.

▶ SHUTDN { [pdisk0] | [ALL] [pdisk1...pdiskn] }

Operator command

The SHUTDN command performs tasks necessary to shutting down the PRIMOS system in an orderly manner.

▶ SIZE pathname

SIZE gives the data size of pathname in records (i.e., the decimal number of records) and words. The number of records in a file is defined to be the number of data words in a file divided by 440, rounded up. The exception to this rule is that a zero word-length file always contains one record.

Example:

```
OK, size shortfile
     1 RECORD IN FILE (41 WORDS)
OK,
```

▶ SLIST [pathname]

SLIST prints (or displays) the contents of a file at the user's terminal.

Pathname specifies the name of a file. If it is a simple filename, it is in the current directory. If **pathname** is omitted, SLIST asks the user to specify a treename (synonym for pathname).

SLIST is often used to display source listings of short programs or data files. If TERM -XOFF has been enabled, the user can halt the SLIST display by typing CONTROL-S, then restart the display by typing CONTROL-Q or quit by typing CONTROL-P.

Examples:

```
OK, slist
Usage: SLIST <treename>
OK, slist shortfile
This file was listed by the SLIST command.
This is the second line of the file.
OK,
```

▶ SORT [-BRIEF] [SPACE] [MERGE] [-TAG / -NONTAG]

The SORT command sorts up to 20 files into a single output file, sorting (in ascending or descending order) on up to 64 keys. SORT is a stable sort: that is, it preserves the order of input for records with equal keys.

SORT's options are as follows:

Option	Meaning
-BRIEF	SORT program messages are not printed at the users terminal.
-SPACE	Any blank lines are deleted from the SORT output file.
-MERGE	A merge of presorted files is requested.
-TAG	A TAG sort (described below) is requested.
-NONTAG	A NONTAG sort (described below) is requested.

A **TAG sort** is specified when large files are sorted. For unordered files, it is a faster sort than non-tag. Internally, the 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.

NONTAG sort may be specified for smaller or well ordered input files. Internally the NONTAG sort stores each input record with its sort key in the work file. This eliminates the search for each record after merging, but requires more disk space.

If **-TAG** or **-NONTAG** are not specified, the system defaults to TAG.

When invoked, SORT prints a message requesting:

- The name of the file to be sorted
- The name of the output file to be created
- The number of keys for the sort (default is 1).

This information should be given on a single line. When it has been supplied, SORT asks for:

- The starting and ending columns of each key field (default is ASCII, "A")
- The data type of the key
- Information on whether the sort on that key is to be done in ascending or descending order. Ascending is the default and need not be specified; descending is indicated by an "R" for reverse.

Information for each key field should be given on a separate line.

If the -MERGE option was given, SORT now asks for:

- The number of additional files to be merged,
- The names of the files.

Give the number and each filename on a separate line.

The dialog for a sample sort is as follows;

```
OK, sort
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
aleph alpha 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 5
8 1Ø ts r

BEGINNING SORT



 PASSES        2          ITEMS         5

[SORT-REV18.Ø]
OK,
```

The same sort, with -BRIEF in force, would be requested by:

```
OK, sort -brief
aleph alpha 2
1 5
8 1Ø ts r

BEGINNING SORT



 PASSES        2           ITEMS         5

[SORT-REV18.Ø]
OK,
```

Several sorted files could then be merged with a mergesort:

```
OK, sort -merge -brief
alpha greek 2
1 5
8 1Ø ts r
2
beta
gamma

BEGINNING MERGE



 PASSES        1           ITEMS         15

[SORT-REV18.Ø]
OK, slist greek
able   49
able   15
able   96-
baker  44
baker  43
baker  14-
charl  95
charl  52
charl  78-
delta  66
delta  44
delta  67-
easy   97
easy   49
easy   12-
OK,
```

CAUTION

Giving identical names for input and output files is not a safe practice. When this is done, 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 the loss of the file being sorted — may result.

File and record types

SORT can process four types of files: ASCII files (also called compressed files), uncompressed files, binary files (also called variable length files), and fixed length files. (All files created by Prime's text editor, ED, are ASCII files.) The file types are defined by the records they contain, as follows:

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	Uncompressed record delimited by a newline character (:212). They cannot contain data that may be interpreted as a newline.
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, it must be included in the character count.

SORT has two default file types: ASCII (uncompressed) and binary (variable length). The type used depends on the type of key selected. If a user specifies any integer or real key, SORT defaults to binary records. Otherwise, the SORT file type defaults to ASCII.

Specifying file and record information

By using whatever keywords are needed from the list below, users can specify multiple input files for SORT or SORT -MERGE, indicate file types, or give information about record length. These keywords replace the standard "inputfile outputfile number-of-keys" line of dialog. They are given on a single line, in any order. All input files must be of one type. Input and output files may be of different types.

Keyword	Usage
-INPUTFILE name	Specifies a file to be sorted. **name** may be a pathname of up to 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	**n** is the number of keys for the sort.
-INTYPE { **COMPRESSED** | **UNCOMPRESSED** | **FIXED** | **VARIABLE** }	Specifies the type of file(s) to be sorted. All input files must be of the same type. If this keyword is not given, a default file type will be taken from the key type.

-OUTTYPE type	Specifies file **type** for the output file. Types are the same as those for input files. If this keyword is not given, the output file will have the same type as the input file(s).
-INLENGTH n	Gives the maximum length of the input records (in bytes). (Greatest possible **n** is 32760 bytes. This is also the default.) This keyword must be given for fixed-length records.
-OUTLENGTH n	Specifies maximum length for records in output file. Defaults to length of input record. If you specify a fixed-length record output file, you must also specify its record length.

Key types

SORT recognizes 13 types of keys. ASCII files (compressed and uncompressed) can be sorted on seven of these key-types: A, LS, TS, LE, TE, and AU. Variable and fixed length files can use any key type.

Table 3-1 explains each key type. Note that the default, A, sorts upper and lowercase characters differently. This represents a change from the Rev. 16 default, which converted lowercase characters to uppercase before sorting them. The lower-and-uppercase-equal sort is now provided by the AU key.

Specifying key information

Key information may also be specified via keywords, as follows:

Keyword	Usage
-START n	**n** is first column of key.
-END n	**n** is last column of key.
-DESCENDING	Requests sort in descending order.
-TYPE code	Any of the codes from Table 2-1.
-EBCDIC	Requests that the EBCDIC collating sequence, rather than the ASCII sequence, be used for sorting (used only with A or AU key types)

An example of a SORT dialog using keywords might be:

```
OK, sort -brief
-input chaos.1 -input chaos.2 -output order -keys 2
1 1Ø
15 2Ø r

BEGINNING SORT



 PASSES        2          ITEMS        16

[SORT-REV18.Ø]
OK,
```

Table 2-1. 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.
I	Single precision integer (short)	Length is 2 bytes; range is -32767 to +32767.
J	Double precision integer (long)	Length is 4 bytes; range is -2**31 to +2**31 -1.
F	Single precision real	Length is 4 bytes; range is ±(10** -38 to 10**38).
D	Double precision real	Length is 8 bytes; range is±(10**-9902 to 10**9825).
U	Numeric ASCII, unsigned	Like plain ASCII. These are stored one digit per byte, and are limited only by the length of the record.
LS	Numeric ASCII, leading separate sign	Numbers preceded by "+" or "–" to indicate positive or negative value. (A blank space will be treated as a positive sign.)
TS	Numeric ASCII, trailing separate sign	Same as LS, except that the "+" or "–" follows the number.
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 (e.g., L579 represents -3579).

Digit	Positive	Negative
0	0,-,+,{	},-
1	1 A	J
2	2 B	K
3	3 C	L
4	4 D	M
5	5 E	N
6	6 F	O
7	7 G	P
8	8 H	Q
9	9 I	R

TE	Numeric ASCII, trialing embedded sign	Same as LE, except that the last digit carries the sign (e.g. 357R represents -3579).
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, this comes out to a full number of bytes. Packed decimal keys may be up to 63 digits plus sign.
AU	ASCII, upper and lower case sort	Storage is identical to regular ASCII. Lowercase characters are sorted as uppercase, put into output file as lowercase.
UI	Unsigned integer	Length is 2 bytes; range is 0 to 65535.

▶ SPOOL [pathname] [options]

The SPOOL subsystem allows any user to submit disk files to be printed or plotted on the system line printers and/or plotter. The subsystem consists of a user interface program (SPOOL), a set of phantom programs to print files queued on the various printing or plotting devices, an operator interface program (PROP), and a queue management facility.

The SPOOL program allows users to submit files to the queue mechanism, interrogate the queue status, and cancel their queued requests. When SPOOL is invoked by a user submitting a file, it responds with a message of the following format:

```
PRT00 spooled,  records:    1, name: SHORTFILE
```

The length of the spool file determines its priority in the queue. Short files are given a higher priority than long ones to ensure efficient queue operation. (To find out what boundary divides short files from long ones on your printers, use the PROP -DISPLAY command.)

The first parameter to give when submitting a file is the **pathname**; subsequent parameters are specified as **options** in the SPOOL command format. Using **options,** the user may request that the spooler provide certain internal services when printing the spool file — for example, inserting line numbers in the left margin, using FORTRAN output conventions (column 1 represents carriage control), etc.

The user may also use **options** to specify other information regarding the spool file: for example, that it be deferred (held in the queue without printing) until a certain time of day, that the file be printed or plotted on a special form type, etc.

Parameter	Meaning or Remarks

pathname	If **pathname** is specified, the file named **pathname** is copied into the SPOOL queue.
	
	If **pathname** is omitted (or if the -OPEN option is used), a file is created in the SPOOL queue. This file has null contents, and it remains open on File Unit 2 (unless otherwise specified by the -TUNIT option). When SPOOL is invoked in this manner, it responds with the message:
	
	`OK, spool -open`
	`[SPOOL rev 18.1]`
	`PRTØØ4 opened.`
	`OK,`
	
	File Unit 2 (or the specified TO file unit) remains open until closed by specific user command or action. When closed, any information placed in the previously open file (such as listings from compilations) is printed, and then deleted, by SPOOL.

-AS alias	The **alias** (a name of 16 or fewer characters) replaces the **pathname** on the file header and in SPOOL -LIST displays.
-AT destination	Denotes the printer (or printers) which may print the file. Useful for specifying location when the computer is connected to others via PRIMENET, or when output is delivered to mailstops.

-CANCEL	Removes the print or plot request(s) (specified by their PRT numbers) from the queue. For example:
	
	`OK, spool -cancel 2, ØØ3, prtØØ4`
	`[SPOOL rev 18.1]`
	`PRTØØ2 cancelled.`
	`PRTØØ3 cancelled.`
	`PRTØØ4 cancelled.`
	`OK,`

-COPIES n	Specifies number of times file is to be printed. (**n** must not 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 specified **time**. The **time** may be entered in either 24 hour format (00:00 = midnight) or in 12 hour format, with AM or PM (12:00 AM = midnight).
-FORM [type]	Specifies a paper **type**. This file is not printed until the system operator indicates that the requested form is mounted on the printer. The **type** argument must be a 1-6 character name and

does not have to start with an alphabetic character. If -FORM is not given, the default paper **type** is used.

-FTN Indicates FORTRAN output conventions are to be used when printing the file.

Characters in column 1 of each line have special significance, as follows:

1	Eject to top of page before printing
0	Skip 2 lines
space	Skip 1 line
+	Overprint last line (available on MPC parallel-interface printers only)
-	Triple space

The spooler will not generate an initial page eject after the banner page is output when this mode is specified.

-LIST Lists all SPOOL -queue entries or selected SPOOL-queue entries. Possible values and associated meanings for list option are as follows:

List Option	Meaning
OWN	Lists files spooled under the current user's LOGIN-name.
DEFER	Lists deferred files
PLOT	Lists files in plot queue
PRINT	Lists files in print queue
FORM type	Lists files queued with the form specified by type. For default form do not specify type.
ALL	Lists all files in queue. This is the default for list option.

-LNUM Generates line numbers. The spooler prefixes each line of output with a four-to-five digit line number enclosed in parentheses and followed by a space. This option may not be used with the FORTRAN print convention option, -FTN. It also overrides carriage control characters.

-NOFMT Indicates no format control. The spooler supplies no format control (pagination and header generation) as the file is printed. It is the user's responsibility to place the necessary line-printer control codes in the text of the file to be printed.

-NOHEAD Prints file without header or trailer pages.

-OPEN Specifies open-only operation. Instead of copying the named file to the spool queue, SPOOL creates a new queue file and opens it on unit specified by -TUNIT.

-PLOT [nwords] Denotes a plot file; **nwords** is the decimal number of words to be read and output per raster scan. If **nwords** is omitted, the default is 128 (for a 200 raster/inch plotter). This spool file is invisible to the line-printer spooler.

-TUNIT unit Defines TO-unit number. This is the file **unit** that is opened on an open-only operation and is used for the output of a copy operation.

The restrictions regarding **unit** conflict and argument specification described with -FUNIT apply also when using -TUNIT.

Multiple options: Any or all of the above options may be used jointly in a single SPOOL command line. For example:

```
OK, spool shortfile -as testl -at 1 -defer 2200
[SPOOL rev 18.x]
PRT004 spooled,  records:    1, name: SHORTFILE
OK,
```

This particular command requests that the file named "O_17" be printed at the "bldg.1" printer, under the alias of "EX.1", at 10 pm (22:00).

If "-LIST" or "-CANCEL" appears on the command line, it must appear as the last option.

See also: **PROP, CONCAT.**

▶ SPSS [-INPUT] [-LISTING] [-SIZE] [-PAGESIZE] [-PRINTBACK] [-MAXERROR] [-EDIT]

The SPSS command starts the Statistical Package for the Social Sciences program under PRIMOS. For detailed information, refer to **SPSS On Prime Computers.**

▶ START [pc] [a] [b] [x] [keys]

START initializes the processor's registers and keys from the command line (or from RVEC, for any values not specified in the command line) and starts execution at location **pc**. This form of the command starts or restarts a static mode program previously loaded into memory by a RESTOR or RESUME command. (Static mode programs include external PRIMOS commands and most user software. Recursive mode programs, discussed below, include internal PRIMOS commands and all on-units.)

START can also restart a static or recursive mode program that has returned control to PRIMOS (for example, because of a BREAK, an error, or a FORTRAN PAUSE or CALL EXIT statement). If START is typed without arguments, one of the following two things occurs:

1. If no static mode program has been invoked or restored at this listener level (as is the case after a BREAK, for example), the interrupted program is continued from the point of interruption, whether it is static or recursive mode.
2. Otherwise, the static mode program defined by the current RVEC is invoked.

If arguments are given, their register settings are applied to the RVEC contents (defining a new static mode machine state), and that static mode program is invoked.

Note

If a static mode program is interrupted, and another static mode program is invoked subsequently, an attempt to START the first program will be refused with an appropriate error message. This is done 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.

▶ STARTUP [PROTECT] pdisk0 [pdisk1...pdisk7] *Operator command*

STARTUP initializes the configuration of disk drives by relating physical disk drive numbers to PRIMOS logical disk unit numbers. Physical device numbers for disks are given in the **System Administrator's Guide.**

See also: **ADDISK.**

▶ **STATUS** [**ALL** | **DISKS** | **USERS** | **NETWORK** | **UNITS** | **ME** | **DEVICE** | **SYSTEM**]

The STATUS command prints information about the condition of PRIMOS. For example, it can show what disks are running on the system, the status of other systems connected via PRIMENET, or who all the other users are.

Options to STATUS:

STATUS ALL	Gives all the information STATUS can provide.
STATUS DEVICE	Gives physical and logical device numbers for any assigned magnetic tape drives.
STATUS DISKS	Shows which disks are running.
STATUS USERS	Shows who the users are.
STATUS ME	Gives user information only about the user who makes the request.
STATUS NETWORK	Shows status of other PRIMOS systems on PRIMENET.
STATUS UNITS	Shows which file units the user has open.
STATUS SYSTEM	Prints the version of PRIMOS currently running.
STATUS (alone)	Combines information of STATUS UNITS, STATUS DISKS, STATUS NET, STATUS SYSTEM, and STATUS ME.

Explanation of STATUS items:

These are all the column headers printed by STATUS ALL, in the order of appearance.

FILE UNIT	The file units that the user has open.
FILE POSITION	Position (in words) within the file.
OPEN MODE	Tells whether file has been opened for reading, writing, or both.
FILE TYPE	Tells if file is SAM file, DAM file, SEGSAM, SEG DAM or UFD.
RW LOCK	Tells how many readers and writers file may have. (N = any number.)
TREENAME	Pathname of the file.
DEVICE	Physical device number of magnetic tape drive.
USRNAM	Name of user to whom device is assigned.
USRNUM	User number of user to whom device is assigned.
LDEVICE	Logical device number of device. LDEVICE differs from DEVICE only if the user has given the device a logical alias when assigning it.
DISK	The volume name of the disk.
LDEV	The logical device number, in octal, of a disk volume.
PDEV	The physical device number, in octal, of a disk volume. It specifies the physical location on a physical disk where the logical disk volume begins.
SYSN	The name of the PRIMENET node on which the disk volume is running. If blank, it means the disk is on the local system.

NODE The name of a PRIMOS system on PRIMENET.

STATE Whether a node is UP, DOWN, or OFFLINE. The node whose state is **** is the local node, i.e. this computer.

USER Each user's login name.

NO The user number, or job number, in decimal.

LINE The number of the line through which the user is connected to PRIMOS. Phantoms are shown as being on line 77, a user at the supervisor terminal as on line 76, and users logged in across PRIMENET as on line 75. An octal number.

DISKS The physical disks and other devices in use by each user. A disk is "in use" if the user's home or current UFD resides on it, or if the user has opened a file on it. Assigned devices are identified by the abbreviations used by the ASSIGN command, except for disks (which are identified by name) and AMLC lines (identified as AL n, where n is the line number).

Examples:

1. **STATUS** with no parameters specified:

```
OK, status

System is currently running PRIMOS rev. 18.1


USR=BEECH  NJK

File   File       Open     File
Unit   Position   Mode     Type   RWlock   Treename
 127   000000080     W      DAM    NR-1W   <FOREST>BEECH>XS.13


DISK    LDEV   PDEV   SYSN
SYS.K    0      460
QAGRP3   1    12060
PLAINS   2    52061
OCEANS   3    22062
FOREST   4    61463
SYS.E    5      460   NJE
QAGRP1   6    12460   NJE
QAGRP2   7    61461   NJE
MFG     10      462   NJE
AETHER  11    21462   NJE
SYS.B   12            NJB
MOUNTS  13            NJB
REEF    14            NJB
SYS.D   15      460   NJD
FARMS   16    12060   NJD
FARMS2  17    52061   NJD
```

```
RING NETWORK

      NODE       STATE
    NJK          ****
    NJB           UP
    NJD           UP
    NJE           UP
    NJC          DOWN

USER    NO LINE DISKS
BEECH   49  REM <FOREST> (FROM NJB    )

OK,
```

2. **STATUS DISKS**

```
OK, status disks


DISK    LDEV  PDEV  SYSN
SYS.K   Ø     46Ø
QAGRP3    1  12Ø6Ø
PLAINS    2  52Ø61
OCEANS    3  22Ø62
FOREST    4  61463
SYS.E     5    46Ø  NJE
QAGRP1    6  12460  NJE
QAGRP2    7  61461  NJE
MFG      1Ø    462  NJE
AETHER   11  21462  NJE
SYS.B    12         NJB
MOUNTS   13         NJB
REEF     14         NJB
SYS.D    15    46Ø  NJD
FARMS    16  12Ø6Ø  NJD
FARMS2   17  52Ø61  NJD

OK,
```

3. **STATUS USERS**

```
OK, status users

USER     NO LINE DISKS
SYSTEM    1  ASR <SYS.K>  AL57
CROW      7    5          (TO NJE    )
PERCH     8    6 <PLAINS>
ELM      11   11          (TO NJB    )
BALSA    13   13          (TO NJB    )
OWL      19   21 <QAGRP3>
HAWK     21   23 <QAGRP3>
CORAL    38   44          (TO NJE    )
WILLOW   46   54 <QAGRP3>
BEECH    49  REM <FOREST> (FROM NJB    )
```

```
PARROT   50  REM <QAGRP3> (FROM NJE   )
BIRCH    51  REM <PLAINS> (FROM NJB   )
FAM      94   PH <QAGRP3> <SYS.K>
SYSTEM   95   PH <SYS.K> (2)

OK,
```

4. **STATUS ALL**

```
OK, status all

System is currently running PRIMOS rev. 18.1


USR=BEECH  NJK

File   File       Open     File
Unit   Position   Mode     Type   RWlock   Treename
 127   000000080     W      DAM    NR-1W   <FOREST>BEECH>XS.16

DEVICE  USRNAM  USRNUM  LDEVICE

DISK    LDEV  PDEV  SYSN
SYS.K    0    460
QAGRP3   1   12060
PLAINS   2   52061
OCEANS   3   22062
FOREST   4   61463
SYS.E    5    460   NJE
QAGRP1   6   12460   NJE
QAGRP2   7   61461   NJE
MFG     10    462   NJE
AETHER  11   21462   NJE
SYS.B   12          NJB
MOUNTS  13          NJB
REEF    14          NJB
SYS.D   15    460   NJD
FARMS   16   12060   NJD
FARMS2  17   52061   NJD

RING NETWORK

      NODE       STATE
    NJK          ****
    NJB           UP
    NJD           UP
    NJE           UP
    NJC          DOWN

USER     NO LINE DISKS
SYSTEM    1  ASR <SYS.K> AL57
CROW      7    5          (TO NJE   )
PERCH     8    6 <PLAINS>
ELM      11   11          (TO NJB   )
```

```
BALSA   13   13          (TO NJB   )
OWL     19   21 <QAGRP3>
HAWK    21   23 <QAGRP3>
CORAL   38   44          (TO NJE   )
WILLOW  46   54 <QAGRP3>
BEECH   49  REM <FOREST> (FROM NJB   )
PARROT  5Ø  REM <QAGRP3> (FROM NJE   )
FAM     94   PH <QAGRP3> <SYS.K>
SYSTEM  95   PH <SYS.K> (2)

OK,
```

5. **STATUS ME**

```
OK, status me

USER    NO LINE DISKS
BEECH   49  REM <FOREST> (FROM NJB   )

OK,
```

6. **STATUS DEVICE**

```
OK, status device

DEVICE  USRNAM  USRNUM  LDEVICE
MTØ     THORN     5     MTØ
```

7. **STATUS UNITS**

```
OK, status units


USR=BEECH  NJK

File   File      Open    File
Unit   Position  Mode    Type  RWlock  Treename
 127  ØØØØØØØ6Ø    W     DAM   NR-1W  <FOREST>BEECH>XS.19


OK,
```

8. **STATUS NET**

```
OK, status net


RING NETWORK

     NODE      STATE
   NJK         ****
   NJB          UP
   NJD          UP
   NJE          UP
   NJC         DOWN


OK,
```

9. **STATUS SYSTEM**

```
OK, status system

System is currently running PRIMOS rev. 18.1


OK,
```

Under PRIMOS II, STATUS lists the login UFD, the logical device upon which the UFD resides, the low boundary of PRIMOS plus buffers. the open file units. and the physical-to-logical device correspondence. STATUS also lists physical device numbers, as described in **The System Administrator's Guide.**

▶ SVCSW {0 / 1}

The SVCSW command controls the handling of SVC instructions in a virtual memory environment.

The normal mode (SVC 0) causes all SVC instructions to be trapped and processed by the system supervisor. 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. The class of SVC instructions always processed by the PRIMOS operating system, regardless of the SVCSW command, are those determined by FUNCTION code 5XX. Currrently the SVC's are RREC, WREC (for reading and writing to disk), TIMDAT (for obtaining the time and date from PRIMOS), and RECYCL.

The SVC switch is initialized to 0 by the LOGIN command.

▶ TCF -HOST hname -TERMINAL tname [-QUIT 'q string']

This command invokes DPTX/TCF (Transparent Connect Facility) on a system where DPTX/TSF and DPTX/DSC are running. The parameters for TCF are as follows:

hname Is the PRIMOS name for the IBM host mainframe the user wishes to connect to.

tname Is the PRIMOS name of the IBM 3270-type terminal from which the user wants to communicate (the current terminal may be specified by '*')

q string Is a **string** of eight characters or less that will serve as the signal that the user wishes to return to PRIMOS command level. The default **string** is 'TCF$QUIT'; it must be given in all caps, as upper and lower case characters are considered different characters when the QUIT string is being evaluated.

Note

When the quit string is entered at the terminal, the program breaks the connection with the host, prints out 'TCF HALT', and returns to PRIMOS. To the host, it seems that the terminal was powered off. Some applications programs may not tolerate this; consult your installation administrator for guidance.

▶ **TERM** (**-ERASE char** | **-KILL char** | **-BREAK ON** | **-BREAK OFF** | **-FULL** | **-HALF LF** | **-HALF NOLF** | **-DISPLAY** | **-NOXOFF** | **-XOFF**)

The TERM command allows the user to define his terminal characteristics. It controls duplex and half-duplex operation, designates KILL and ERASE characters, and enables or disables the BREAK (CONTROL-P) key.

The TERM command with a null argument prints or displays a list of the optional parameters at the terminal.

Multiple options may be specified on one line with the TERM command. For example:

```
term -half -xoff
```

may be specified on one line. See below for definitions of -HALF and -XOFF.

Parameter	Meaning or Remarks
-ERASE char	Sets the ERASE character to the character **char**, for the duration of the user's process. The default ERASE character, set upon LOGIN, is the double-quote (").
-KILL char	Sets the KILL character to the character **char**, for the duration of the user's process. The default KILL character is the question mark (?).
-BREAK OFF	Inhibits the CONTROL-P or BREAK character from interrupting a running process. The default condition, 'BREAK ON', is set by the LOGIN command, and allows CONTROL-P or BREAK to interrupt a process.
-BREAK ON	Enables the CONTROL-P, or BREAK, character to interrupt a running process.
-FULL	Sets PRIMOS to treat the user terminal as a full-duplex terminal. Under PRIMOS, normal operation is full-duplex. This option is not reset to a default by the LOGIN command; therefore the user must reset it. With full-duplex operation, all characters are echoed except a Line-Feed input character, which is ignored. A Carriage-Return input character is passed into the system as a Line-Feed and echoed to the user as Carriage-Return followed by Line-Feed.
-HALF [LF]	Sets PRIMOS to treat the user terminal as a half-duplex terminal. No characters are echoed except Carriage-Return, which echoes a New-Line. Input is sent to the Line Feed system in the same manner as with TERM -FULL. LOGIN does not reset the terminal to a default setting of half duplex or full duplex. Therefore, the user must reset to half- or full-duplex, as desired.
-HALF NOLF	Functions in the same manner as -HALF except that the Carriage Return does not echo as Line Feed.

-DISPLAY	Prints (or displays) the current values for the ERASE and KILL characters at the user terminal.
-XOFF	Sets the terminal to full-duplex (default value) and enables the X-OFF (CONTROL-S) and X-ON (CONTROL-Q) keys.
-NOXOFF	Sets the terminal to full-duplex (default value) and disables the X-OFF (CONTROL-S) and X-ON (CONTROL-Q) keys.
-XOFF -HALF	Half duplex with X-OFF enabled.

Note

The X-OFF feature is not available under PRIMOS II or III.

TIME

The TIME command prints the current value stored in the PRIMOS time accounting registers. The three values printed are the same as the three values in the logout message:

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.

The disk I/O time includes user-requested I/O to files and also paging I/O time generated on the user's behalf. All times include 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. Compute time does not include I/O time for diskette or for disks connected to a type 4000 Controller.

Example:

```
OK, time
  Ø:35   Ø:23   Ø:14
OK,
```

TRAMLC

TRAMLC transmits or receives a file over an assigned AMLC line between two Prime computer systems using transparent protocol.

TRAMLC's first prompt is:

```
FNAME
```

Reply by giving the filename or pathname of the file you wish to transmit or receive. TRAMLC then prompts:

```
T/R  LINE#  BLOCK
```

This prompt is explained as follows:

T/R	Transmit or receive? Answer with T if you want to send a file, R if you want to receive one.
LINE#	Reply with the number of your AMLC line.
BLOCK	When TRAMLC sends a message, it divides it into 60-word blocks and transmits one block at a time. If you want to monitor the progress of your transmission, TRAMLC will send messages to your terminal, saying "Block n transmitted." The number of messages you receive depends on the response you give to this prompt. If your response is "0", TRAMLC sends no messages. If your response is a positive integer (n), TRAMLC sends a message at every n blocks.

When the entire file has been transmitted or received, the message

```
FILE COMPLETE
```

is printed at the user's terminal.

Either the transmitter or receiver program can be started first. Both devices must be running at the same baud rate.

Note

TRAMLC is designed for conveniently transmitting/receiving small files between two Prime computers. For maximum performance reasons, TRAMLC should not be used as a communications link.

Error messages give the reason for the error that occurred and the block number of the failure.

▶ TYPE text

The TYPE command prints text at the user's terminal or into a command output file.

text contains the message or information to be displayed. **text** may also contain variable references and function calls. Specifying variable references and function calls is more useful within CPL programs than at command level.

▶ UNASSIGN device

The UNASSIGN command may be entered at the user terminal (to which a **device** is currently ASSIGNed) or at the supervisor terminal. The effect of this command is the opposite of ASSIGN. The UNASSIGN command, entered at the system terminal, unconditionally deassigns the peripheral assigned to any user. Entered from a user terminal, UNASSIGN deassigns 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.

device is a previously assigned device, as defined in the ASSIGN command.

From the supervisor terminal, this command can be used to release a **device** if the user who assigned it has forgotten to log out and has left his terminal.

Example:

```
UN PTR
```

unassigns the paper-tape reader.

If an operator UNASSIGNs a user-dedicated tape drive, no message will appear at that user's terminal. Should the user subsequently attempt to unassign the same device, an error message will be displayed.

Magnetic tapes can be unassigned by physical or logical device number:

```
UNASSIGN MTpdn

UNASSIGN -ALLIAS MTldn
```

The -ALIAS option can be specified only if a logical device number was previously assigned to this particular drive. UNASSIGN returns only the "OK," prompt to the terminal from which it is issued.

For a disk to be assigned to a user as an I/O device, it must not be assigned to another user nor be in the PRIMOS file system. It must also be specified by an entry in the Assignable Disks Table (refer to ASSIGN). If the disk is assigned to PRIMOS, it must be released by the SHUTDN com-

mand at the supervisor terminal. A disk that has been ASSIGNed by a user cannot be entered as an argument in the ADDISK command. The supervisor terminal can UNASSIGN any device that may be assigned. All devices ASSIGNed by a user are released when the LOGOUT command is invoked by that user.

▶ UPCASE in-pathname out-pathname

UPCASE reformats files that contain lowercase alphabetic characters, making them suitable for output to a device with only uppercase alphabetic characters. UPCASE scans through an input file, replacing all occurrences of lowercase characters with their counterparts. **in-pathname** is the input file and **out-pathname** is the output file.

▶ USERS

The USERS command prints the number of users currently logged into PRIMOS.

Example:

```
OK, USERS

USERS= 24

OK,
```

▶ USRASR userno *Operator command*

The USRASR command allows the supervisor terminal to act as a user terminal. The parameter **userno** is the user number to be associated with the supervisor terminal.

▶ VPSD, VPSD16

VPSD is the V-mode version of PSD. It is described in the **Assembly Language Programmer's Guide**.

VPSD16 is a version of VPSD that is loaded at 160000 octal. Its commands are the same as those for VPSD.

▶ VRTSSW [number]

The VRTSSW command sets the virtual sense switches. There are 16 switches. **number** is an octal representation of them as a 16-bit word; the default is 0. It is stored and made available to the user when a program written in PMA executes an INA '1620 (read sense switches) instruction. The virtual sense switches are initialized to 0 by LOGIN.

Example:

```
V 40100
```

Note

The Skip-On-Sense-Switch series of instructions always refers to the actual sense switches, not the virtual sense switches.

▶ WS1004
WS200UT
WS7020
WSX80
WSGRTS
WSHASP

These commands control the Remote Job Entry workstations (that is, the terminals that control RJE operations). The systems emulated are:

WS1004	Univac 1004
WS200UT	CDC 200 UT
WS7020	ICL 7020
WSX80	IBM2780 and 3780
WSGRTS	Honeywell GRTS
WSHASP	IBM HASP

For further information on these commands, see the **Remote Job Entry Guide.**

▶ $$

The $$ command is used in command files to flag information to be passed to the Batch monitor. Its use is explained in the writeup on JOB.

▶ /* [Any-text-desired]
*[Any-text-desired]

A command line that begins with an asterisk (*) or a slash-asterisk (/*) is interpreted as a comment line; no action is taken.

Example:

```
/*  This is the last example in this section.
```

3

FUNCTIONS AND VARIABLES

This section discusses the use of global variables and command functions at command level. It also lists and describes all Prime's command functions.

USING VARIABLES AT COMMAND LEVEL

There are four commands that govern the use of variables.

- **DEFINE_GVAR** creates or activates a file that stores global variables.
- **SET_VAR** defines individual variables and places them (with their values) inside an active global variable file.
- **LIST_VAR** displays global variables and their values.
- **DELETE_VAR** removes one or more variables from an active global variable file.

Each of these commands is explained in detail in Section 2. An example of their use is as follows:

1. A user decides to create a global variable file and name it VARFILE. She creates it with the command:

```
OK, DEFINE_GVAR VARFILE -CREATE
```

2. She places two global variables in the file, using the SET_VAR command:

```
OK, SET_VAR .HOME BEECH>BRANCH5>TWIG3
OK, SET_VAR .AWAY BEECH>BRANCH2>TWIG4
```

3. She checks the variables with the LIST_VAR command:

```
OK, LIST_VAR
.AWAY                                   BEECH>BRANCH2>TWIG4
.HOME                                   BEECH>BRANCH5>TWIG3
```

4. She uses the variables in a command:

```
FTN %.HOME%>FOO.FTN -B %.AWAY%>FOO.BIN
```

When the user logs out, her global variable file becomes inactive; but the variables remain within it. When the user logs in again, she can reactivate her file, use her variables, change their values, define new variables, and so on.

Global variables may be used:

- At command level
- Inside a CPL program
- Inside a user program

For further details, see **The CPL User's Guide.**

COMMAND FUNCTIONS

Command functions are intended primarily for use with Prime's Command Procedure Language, CPL. However, they may be invoked at command level as well.

Commands and command functions

When invoked the function and its arguments are placed inside square brackets. For example:

```
OK, TYPE [CALC 254 * 19]
4826
OK,
```

As this example shows, function calls at command level are most often used as arguments to commands.

Types of command functions

Command functions fall into four categories:

Type of Function	Functions	Uses
Arithmetic Functions	CALC, HEX, MOD, OCTAL, TO_HEX, TO_OCTAL	Arithmetic operation and conversions.
File Structure Functions	ATTRIB, DIR, ENTRYNAME EXITS, OPEN_FILE, PATHNAME, READ _FILE. WILD, WRITE_FILE	Identify a file's type or length, verify its existence, retrieve directory names, open files, read or write text into opened files.
String Handling Functions	AFTER, BEFORE, INDEX, LENGTH, NULL, SUBST, SUBSTR, QUOTE, SEARCH, TRANSLATE, TRIM, UNQUOTE, VERIFY	Locate, print, remove or replace characters in a string; translate lowercase to uppercase; add or remove quotes around strings.
Miscellaneous Functions	CND_INFO, DATE, GET_VAR, QUERY, RESCAN, RESPONSE	Query the condition mechanism, print current date and time, get a variable's value, ask for terminal input, unquote and evaluate an expression.

The rest of this section explains these functions in detail.

ARITHMETIC FUNCTIONS

▶ [CALC expression]

The CALC function evaluates arithmetic and logical **expressions.** It accepts **expressions** containing the logical operators & (and), | (or), and ^ (not); the arithmetic operators +, -, *, /, unary +, and

unary -; and the relational operators =, <,> , <=,> =, and ^ =. The precedence is:

```
Highest: ^  unary + unary -
          / *
         + -
         = ^ = < > <= >=
         &
Lowest:  |
```

Note

All operators that are to be 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.

Logical and relational operators return the Boolean values TRUE and FALSE. For example:

```
[CALC 9 > 3]
```

returns:

```
TRUE
```

Relational operators accept either numeric or non-numeric operands. If a relational operator is given a non-numeric operand, a string comparison will be done. If both operands are either numeric or Boolean, an arithmetic comparison is done. Boolean true is interpreted as "1" and false as "0".

Arithmetic operators *must* have operands that convert to integers. (Strings that convert to integers must contain only digits, except possibly for a preceding sign and leading and trailing blanks: the resulting value must be in the range -2**31 + 1 ... 2**31 -1.)

Arithmetic operators return a character string representation of the numeric result. Arithmetic operators apply only to integer values; CPL has no floating point arithmetic.

All the arithmetic operators have the usual definition, except for / which returns only the truncated integer part of any non-integer result. The final result is converted to a string and that string is returned as the value of CALC. For example:

```
[CALC 18 / 5]
```

returns:

```
3
```

▶ [HEX number]

The HEX function converts a hexadecimal **number** to its decimal equivalent.

number is the hexadecimal number or letter you want converted. For example:

```
OK, TYPE [HEX A]
1Ø
OK,
```

The number 10 is returned as the decimal equivalent of A. Negative numbers are not supported.

▶ [MOD string 1 string 2]

The MOD function divides one number by another and returns the remainder (modulus).

string-1 is the dividend **string-2** is the modulus. For example:

```
OK, TYPE [MOD 1Ø 123]
1Ø
OK, TYPE [MOD 36Ø 23]
15
OK,
```

▶ [OCTAL number]

The OCTAL function converts an octal number to its decimal equivalent.

number is the octal number you want converted. For example:

```
OK, TYPE [OCTAL 1Ø]
8
OK,
```

8 is returned as the decimal equivalent of octal 10. Negative octal numbers are not supported.

▶ [TO_HEX number]

The TO_HEX function converts a decimal number to its hexadecimal equivalent.

number is the decimal number you want converted. For example:

```
OK, TYPE [TO_HEX 15]
F
OK,
```

F is returned as the hexadecimal equivalent of decimal 15.

▶ [TO_OCTAL number]

The TO_OCTAL function converts a decimal number to its octal equivalent.

number is the decimal number you want converted. For example:

```
OK, TYPE [TO_OCTAL 8]
1Ø
OK,
```

The number 10 is returned as the octal equivalent of the decimal number 8.

FILE SYSTEM FUNCTIONS

▶ ATTRIB pathname { -TYPE | -DTM | -LENGTH }

The ATTRIB function returns information about a specified file or directory.

pathname is the name of the file or directory. One of the control arguments **(-TYPE, -DTM, -LENGTH)** must be specified.

The **-DTM** option returns the date /time modified information on the file in the format produced by [DATE -full]. The **-LENGTH (-LEN)** option returns the length of the file in words.

If **-TYPE** is specified, ATTRIB returns the filesystem type of **pathname**

- **SAM**, for a sequential access file (such as those created by the EDITOR, ED)
- **DAM**, for a direct access file
- **SEGSAM**, for a segmented sequential access file

- **SEGDAM**, for a segmented direct access file
- **DIR**, for a directory (UFD)
- **UNKNOWN** if a file is not of a recognized type.

▶ [DIR pathname]

The DIR function returns the directory part of **pathname**. That is, all of **pathname** except its final component. For example:

```
[DIR SMITH>X>Y]
```

returns:

```
SMITH>X
```

"*" is returned if **pathname** contains no >'s.

▶ [ENTRYNAME pathname]

The ENTRYNAME function returns the entryname portion of **pathname**. That is, its final component. For example:

```
[ENTRYNAME SMITH>X>Y]
```

returns:

```
Y
```

pathname itself is returned if it contains no >'s.

▶ [EXISTS pathname [type]]

The EXISTS function is a Boolean function that determines:

- whether or not a file system object exists
- whether **pathname** is a specified type (file, directory, segment directory)

pathname is the name or **pathname** of a file or directory.

type may be omitted or may be one of the following:

- **-ANY**
- **-FILE**
- **-DIRECTORY** or **-DIR**
- **-SEGMENT_DIRECTORY** or **-SEGDIR**

The value TRUE is returned if **pathname** is found and it 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 a UFD contains three files: PAYROLL.COBOL, COMPILE_ALL.CPL and PHONE_LIST. Assume also that it contains two sub-UFDs, WORKFILES, and MEMOS. If you were attached to this UFD, the function call:

```
OK, TYPE [EXISTS PHONE_LIST -FILE]
```

would return:

```
TRUE
OK,
```

because PHONE_LIST is a file in the current directory. The function call:

```
[EXISTS MEMOS -SEGDIR]
```

would return:

```
FALSE
```

because MEMOS is not a segment directory.

But:

```
[EXISTS MEMOS]
```

or:

```
[EXISTS MEMOS -ANY]
```

returns:

```
TRUE
```

▶ [OPEN_FILE pathname status-var -MODE m]

The OPEN_FILE function opens a file for reading or writing. It returns the unit number of the file unit on which it opened the file.

pathname is the name or pathname of a file or directory.

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. **Status-var** *must* be specified and must be a global variable if OPEN_FILE is used at command level.

-MODE m indicates how a file is to be opened; reading only, writing only, or reading and writing. If **mode** is omitted, the file is opened for reading.

m is 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*

▶ [PATHNAME path]

The PATHNAME function returns the full pathname (complete with volume name) of **path**. If **path** is *, PATHNAME returns the full pathname of the home directory. The directory specified by the directory part of **path** must exist, and the user's process must be able to attach to it. This is the only way the full pathname can be obtained.

For example:

```
[PATHNAME *>X>Y]
```

returns:

```
<VOLXX>USER>FILES>X>Y
```

if the home directory is <volxx>user>files.

▶ [READ_FILE unit status-var]

The READ_FILE function reads a line from the ASCII file opened on **unit** and returns the line as its value. The value is quoted if it contains special characters. The true null string is returned if end of file is reached.

unit is the decimal number (integer only) identifying the file to be read. It may be a variable whose value was set by the OPEN_FILE function.

status-var is a variable that is automatically set to 0 if the operation is successful and nonzero otherwise. If the end of file is reached, **status-var** will be set to 1.

▶ [WILD wild-name-1 [...wild-name-n] [options]

The WILD function produces a list of all names within a directory that match one or more wildcard names. It has two forms, discussed below. The first form returns all matching names at once, in a single list. Names within the list are separated by blanks. 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 will match. 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.) For example:

```
ATTACH MYUFD
TYPE [WILD @.COBOL @.PMA]
```

prints a list of all COBOL and PMA source files in MYUFD.

options represents one or more optional control arguments. These place limits on the objects matched by the specified wildcard names. **options** are as follows:

Option	Meaning
-AFTER date	Select only objects created or modified after the date specified by **date**. (This information is stored as the file's DTM, "date and time modified." Its format is mo /da /yr.)
-BEFORE date	Select only objects created or last modified before the specified **date**.
-DIRECTORIES	Select only directories.
-FILES	Select only files.
-SEGMENT_DIRECTORIES	Select only segment directories.

The second form of the WILD function returns names one at a time, rather than as a list. Its form is:

[WILD wild-names [options] -SINGLE unit-var]

See **The CPL User's Guide** for details on using this form of the WILD function.

▶ [WRITE_FILE unit text]

The WRITE_FILE function writes the contents of **text** into the ASCII file opened on **file-unit text** must be quoted if it contains special characters. One level of quoting is stripped prior to writing.

unit is the decimal number (integer only) identifying the file to be written.

text is the information you want written. It is written as a single line in the file; that is, the newline is automatically added.

STRING HANDLING FUNCTIONS

▶ [AFTER string find-string]

The AFTER function 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, if you had the string ABCDE and wanted to list all characters appearing after D you would say:

```
OK, TYPE [AFTER ABCDE D]
E
OK,
```

The system returns the character E as that character appears after ABCD. **string** can also be a global variable.

▶ [BEFORE string find-string]

The BEFORE function prints all text or characters that appear before the specified text or characters.

For example, if you had the **string** ABCDE and wanted to list all characters appearing before the character C you would say:

```
OK, TYPE [BEFORE ABCDE C]
AB
OK,
```

The system returns the characters AB as they appear before the character C.

▶ [INDEX string find-string]

The INDEX function 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: if you wanted the position number of the characters DE in the string ABCDE you would say:

```
OK, TYPE [INDEX ABCDE DE]
4
OK,
```

The system returns 4 as the string DE occupies the fourth and fifth positions in the string ABCDE.

▶ [LENGTH string]

The LENGTH function prints the number of characters in a given character string.

For example:

```
OK, TYPE [LENGTH This is a test]
14
OK,
```

The text line **This is a test** contains 14 characters, including embedded blanks.

▶ [NULL string]

The NULL function 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.

▶ [SUBST string-1 string-2 string-3]

The SUBST (substitute) function replaces text in one string with text from another.

string-1 is the string to be modified, **string-2** is the string characters or text in **string-1** to be changed and **string-3** is the text or characters replacing **string-1. string-2** and **string-3** are taken whole, not interpreted character by character as in TRANSLATE (see TRANSLATE). Replacement occurs at all places **string-2** is found. For example:

```
OK, TYPE [SUBST aabbaabbaa bb QR]
aaQRaaQRaa
OK,
```

All 'bb's' in string one were replaced with QR.

▶ [SUBSTR string start-position num_chars]

The SUBSTR (substring) function identifies and prints a substring of specified length beginning in one specified position.

For example: If you had the string ABCDE and wanted to locate C and D you would say:

```
OK, TYPE [SUBSTR ABCDE 3 2]
CD
OK,
```

SUBSTR prints out the two characters beginning in the third position. In this case C and D are output.

▶ [QUOTE string1 string2...stringn]

The QUOTE function places an outer pair of quotes around the text specified in **string** and places single quotes around the text within **string**. The 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'
```

```
[QUOTE 'abc ''de'' fg' ]
```

returns:

```
'''abc ''''de'''' fg'' '
```

▶ [TRANSLATE string-1 string-2 string-3]

The TRANSLATE function replaces characters or text in one string with characters or text from another.

string-1 is the text or characters to be modified, **string-2** holds the characters to replace **string-1** with; and **string-3** holds the characters to be removed.

TRANSLATE looks in **string-1** for each character in **string-3**. If it finds the character, it replaces the character with the corresponding character from **string-2**. That is, if char(i) in **string-1** = char(j) of **string-3**, then char(i) of **string-1** is replaced by char(j) of **string-2**.

For example, if you specify:

```
OK, TYPE [TRANSLATE abc 345 cba]
```

The result would be

```
543
OK,
```

For example:

```
OK, TYPE [TRANSLATE 'a mixxpelled xtring' s x]
a misspelled string
OK,
```

If **string-2** and **string-3** are omitted all text in **string-1** is converted to uppercase.

▶ [TRIM string { -LEFT / -RIGHT / -BOTH } char]

The TRIM function removes characters or text from the left, the right or both sides of a specified **string.**

string is the string of characters or text to be modified, and **char** is the characters or text to be removed from string. For example, if you wanted to remove all B's from the string

```
BBBBABCBBB
```

you would say:

```
OK, TYPE [TRIM BBBBABCBBB -both B]
ABC
OK,
```

All occurrences of B to the left and right of ABC are removed.

▶ [UNQUOTE string]

The UNQUOTE function removes the outer pair of quotes around the text specified in **string** and changes all other remaining quotes within **string** to single quotes. For example:

```
[UNQUOTE '''xx''''yy''' ]
```

returns:

```
'xx''yy'
```

▶ [VERIFY s1 s2]

The VERIFY function returns the number (beginning with 1) of the first character in **s1** that does not appear in string **s2**. For example:

```
[VERIFY 1298s8 Ø123456789 ]
```

returns:

```
5
```

as S, the fifth character of 1298S8, does not appear in 0123456789. If all characters of **s1** appear in **s2**, VERIFY returns 0.

MISCELLANEOUS FUNCTIONS

▶ [CND_INFO control_flag]

The CND_INFO function allows a CPL condition handler to examine the condition information

of the most recent condition on the stack. The function returns information requested by the argument specified in **control_flag. Control_flag** must be exactly one of the following:

-NAME	Returns the name of the condition. If no condition frame is on the stack, $NONE$ is returned.
-CONTINUE_SW **or** **-CONT_SW**	Returns the Boolean value of the continue-to-signal switch. If no condition frame is on the stack, FALSE is returned.
-RETURN_PERMIT **or** **-RET_PMT**	Returns the Boolean value of the returned permitted switch. If no condition frame is on the stack, FALSE is returned.

For a complete discussion, see **The PRIMOS Subroutines Reference Guide.**

▶ [DATE [option]]

The DATE function returns the current date and/or time in a variety of formats. If, **option** is omitted, the date only is returned: 03/06/80. The other possibilities are:

Option	Format
-FULL	80-03-06.13:24:49.Thu
-USA	03/06/80
-UFULL	03/06/80.13:24:48.Thu
-DAY	6
-MONTH	March
-YEAR	1980
-TIME	13.24.48
-AMPM	1:24 PM
-DOW	Thursday
-CAL	March 6, 1980
-TAG	800306
-FTAG	800306.132448

▶ [GET_VAR expr]

The GET_VAR function returns the value of the variable name entered in **expr**. The string $UNDEFINED$ is returned if the variable name in **expr** has not been defined. The function is useful for testing if a variable has been set. If GET_VAR is used at command level, only global variables may be used. For example:

```
[GET_VAR .undefined_var]
```

returns:

```
$UNDEFINED$
```

as the variable, undefined_var has not been defined.

▶ [QUERY text [default]]

The QUERY function prints the contents of **text** on your terminal screen and follows it with a question mark.

text is whatever information you want printed. If **text** is null, printing is suppressed. **text** and **default** must be quoted if they contain special characters. One level of quoting is stripped before printing. After **text** has been printed, the system awaits an answer of yes, y, ok, no, n. Upper or lowercase characters may be used. QUERY returns TRUE if an answer of yes, y or ok was supplied or FALSE if your answer was no or N. If you enter a carriage return, **default** becomes the value of QUERY.

If the value of **default** is not supplied, FALSE is assumed.

▶ [RESCAN string]

The RESCAN function removes one level of quotes from **string** and evaluates any function calls or variable references no longer appearing in quotes. For example:

```
[RESCAN '[BEFORE '' [do not eval this] xxx'' x ] ' ]
'[do not eval this]'
```

returns:

```
OK,
```

The function may be used to force evaluation of quoted variable references and function calls.

▶ [RESPONSE text [default]]

The RESPONSE function prints the contents of **text** on your terminal screen and follows it with a colon.

text is whatever information you want printed. If **text** is null, printing is suppressed.

If **text** and **default** contain special characters, they must be enclosed in quotes. One level of quotes is stripped from **text prior to printing.**

After text has been printed, RESPONSE waits for user input. The input is then returned as the value of the function. If a null response is entered, the value **default** is returned. If **default** is omitted, the null string is assumed.

▶ [SEARCH s1 s2]

The SEARCH function returns the number (beginning with 1) of the first character in **s1** that appears in string **s2**. For example:

```
[SEARCH abc.def <>.+ ]
```

returns:

```
4
```

as . is the fourth character in string abc.def. If no character in **s1** appears in **s2**, SEARCH returns 0.

FILE UTILITY

INTRODUCTION

FUTIL is a file utility whose subcommands permit the user to copy, delete, and list files and directories on a per-file or directory basis. An overview of the commands available under FUTIL is shown in Figure 4-1. FUTIL has an ATTACH subcommand that allows attaching to subdirectories by giving a pathname. FUTIL operates with files within User File Directories and segment directories. For complex or repetitive operations, FUTIL may be run from a command file (via the COMINPUT command) or a CPL program.

FILE STRUCTURE

Figure 4-2 is a simple example of a file structure. The PRIMOS file structure on any disk pack is a **tree structure** where the MFD is the root or trunk of the tree, the links between directories and files or subdirectories are branches, and the directories and files are nodes. A **directory tree** consists of all files and subdirectories that have their root in that directory. In Figure 4-2, the directory tree for UFD1 is circled. In FUTIL, an MFD directory **pathname** consists of a list of directories and passwords necessary to move down the tree from the MFD to any directory. A **pathname** is a syntax convention that allows the specification of any file in any directory or subdirectory. For example, the MFD directory pathname for SUFD1 is:

MFD PASSWORD > UFD1 password > SUFD1 password

The greater-than character (>) separates directories in the pathname and suggests that one is proceeding down a tree structure.

Note

File and directory names can be as long as 32 characters. However, passwords can be, at most, six characters long.

An MFD directory pathname may omit the MFD and include the logical disk number of the pack or the packname.

Examples:

```
UFD1 PASSWORD > SUFD1 PASSWORD
< 1 > UFD1 PASSWORD > SUFD1 PASSWORD
< TSDISK > UFD1 PASSWORD > SUFD11 PASSWORD
```

The **logical disk** number may optionally follow the first UFD as follows:

```
UFD1 PASSWORD 1 > SUFD1 PASSWORD
```

If no **packname** or **disk number** is given, the **logical disk** referred to is the lowest numbered logical disk within an MFD in which UFD1 appears. A user, by means of the ATTACH or PRIMOS LOGIN command, may specify a particular User File Directory in the file structure as the home UFD. Additional FUTIL ATTACH commands may refer to either the MFD or any UFD as the starting point. If the logical disk name is specified as *, the MFD of the current logical disk is scanned for UFD1. Names of this form are referred to as **absolute pathnames** (or absolute tree-

names), because the location of the home UFD is not part of the name. A home UFD directory treename consists of a list of directories and passwords necessary to move down the tree from the home UFD to any directory that has the home UFD as the root. For example, if the home UFD is UFD1, the home UFD pathname to SUFD11 is:

***>SUFD11 sufd11password**

"*" represents the home UFD. The home UFD treename to UFD1 is simply *. This form of pathname is also referred to as a **relative pathname.**

DIRECTORY DESCRIPTIONS

User File Directory

A User File Directory (UFD) is a file that consists of a header and an unlimited number of entries. Each entry consists of a 1- to 32-character filename (or sub-UFD name), protection attributes of the file and a disk record address pointer to the file or sub-UFD. On old partitions, names of entries can be, at most, six characters long. Sub-UFDs have the same format as UFDs.

Segment Directory

All users are familiar with UFDs and sub-UFDs through the operation of LOGIN and ATTACH. There is, in addition, another type of directory, called segment directory. A segment directory is a file consisting of as many as 65,535 entries. Each entry in a segment directory is a record address pointer to the first record of a file or another segment directory. A 0 pointer (null pointer) indicates no file at that entry. To refer to a particular file in a segment directory, a user must specify the entry number that points to the desired file. This entry number is specified by the user as an unsigned entry number enclosed in parentheses as in: (185). For example, a treename pointing to a file in a segment directory would have a format such as:

```
MFD PASSW1 > SEGDIR (185)
```

In FUTIL, arguments to the commands are either User File Directory filenames or segment directory filenames, depending on the directory type the file is under.

FUTIL provides a number of commands for copying, deleting or protecting directories and the files within them. Figure 4-3 shows an overview of FUTIL's COPY, DELETE, and PROTECT commands and their functions.

INVOKING FUTIL FROM PRIMOS

To invoke FUTIL, input the command name FUTIL. When loaded, FUTIL prints the prompt character, >, and awaits a command string from the user terminal. To terminate long operations such as LISTF, type CONTROL-P and give the command FUTIL again.

The erase character " and the kill character ? may be used to modify the command string, as in other operating system commands.

FUTIL SUBCOMMANDS

FUTIL subcommands are described in the rest of this section in alphabetical order. Subcommand formats follow the same conventions as PRIMOS commands — abbreviations are shown in **rust**, optional arguments are in brackets, and an ellipsis (...) signifies that the previous element may be repeated.

Many FUTIL commands are significantly affected by the current value of the FROM and TO directories. For an explanation of FROM and TO directories, refer to the description of the FROM and TO commands in this section.

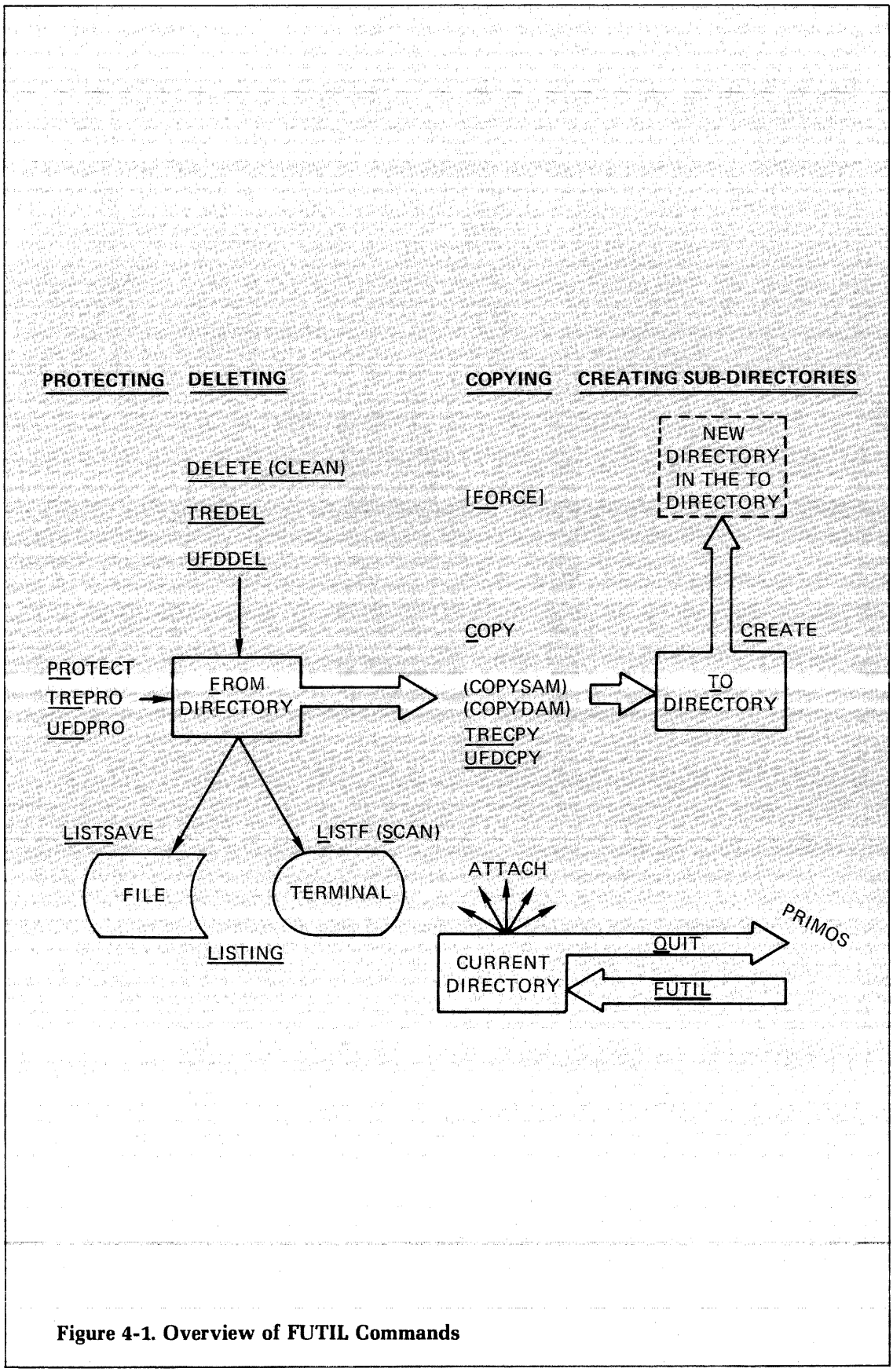


Figure 4-1. Overview of FUTIL Commands

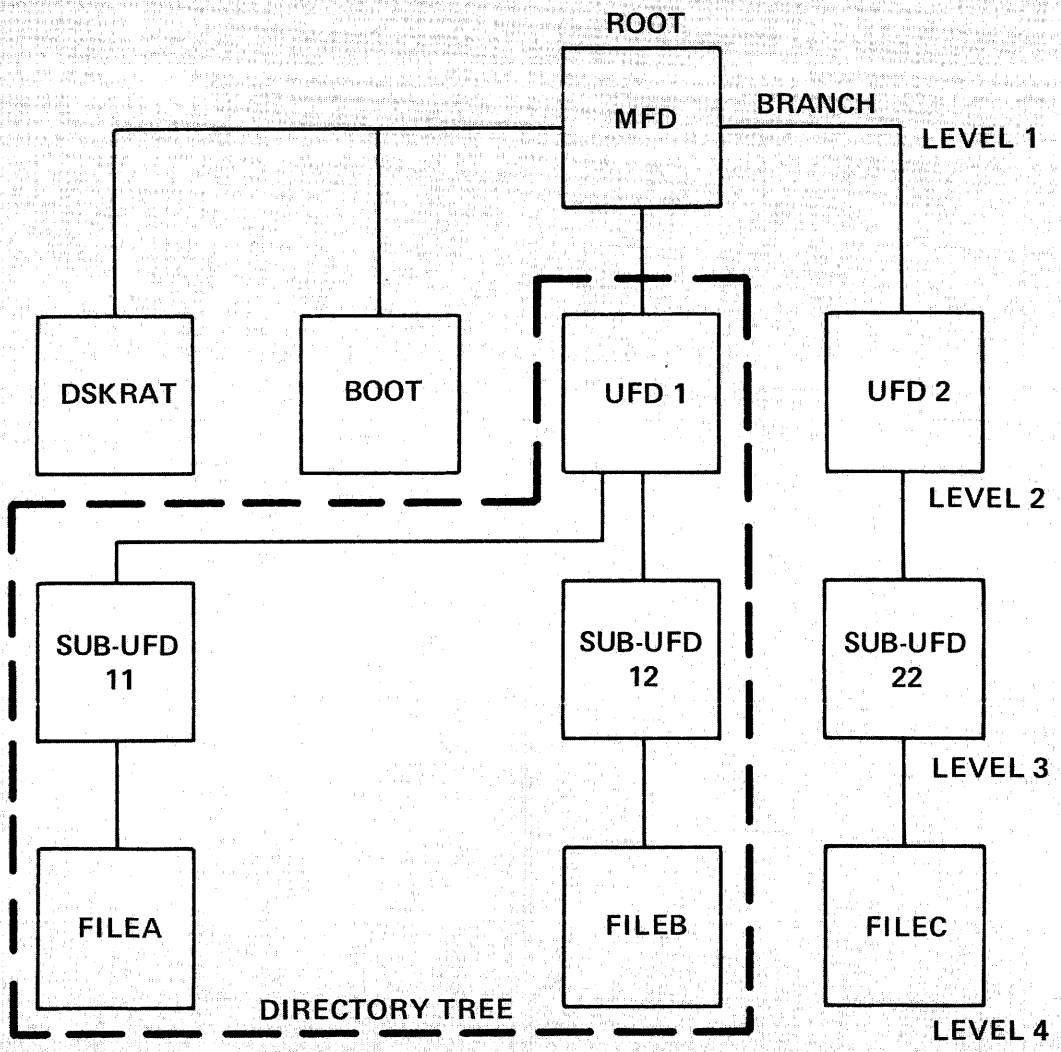


Figure 4-2. Sample File Structure

▶ ATTACH directory-pathname

ATTACH moves the home UFD to the directory defined by **pathname**. The **pathname** may contain, at most, 10 directories. The first directory in the **pathname** may be * (home UFD). All directories in the **pathname** must be UFDs or sub-UFDs. If segment directories are specified within the treename, a BAD STRUCTURE error is reported, and the home UFD is set to the last UFD that was specified in the treename before the error occurred. A subsequent ATTACH command will reset to the TO and FROM names relative to the home UFD (*), but will not affect absolute names.

▶ CLEAN prefix [level]

The CLEAN command is a conditional-delete command based upon a **prefix** match. If a filename begins with the characters specified as **prefix**, the file is deleted. If **level** is specified greater than 1, that many levels of sub-UFDs (including the home UFD) are scanned for **prefix** matches. In no case does CLEAN delete a UFD, sub-UFD, or a segment directory. In the sample tree structure of Figure 4-2, the command: CLEAN F 10 will delete FILEA, FILEB, and FILEC since they all begin with F. A typical usage of CLEAN is:

```
CLEAN L_
CLEAN B_
```

to delete binary and listing files from a UFD.

▶ COPY filea [fileb] [,filec [filed]]...

COPY copies **filea** in the FROM directory to **fileb** in the TO directory and, optionally, **filec** in the FROM directory to **filed** in the TO directory, etc. Filename pairs must be separated by commas. If the second filename of a pair is omitted, the new file is given the same name as the old file. The files **filea, filec**, etc. must be SAM or DAM files and cannot be directories. Read access rights are required for **filea** and **filec**. If **fileb** exists prior to the copy, it must be a SAM or DAM file and the user must have read, write, and delete/truncate access rights to the target file (fileb in this case). If **fileb** exists, it is deleted; then **filea** is copied to **fileb**. The file type of **fileb** will be the same as **filea**.

Examples:

```
COPY FILEA
```

copies FILEA in the FROM directory to FILEA in the TO directory.

```
COPY FILEA , FILEB , FILEC
```

copies FILEA, FILEB, and FILEC in the FROM directory to FILEA, FILEB, and FILEC in the TO directory.

```
COPY FILEA FILEB
```

copies FILEA in FROM directory to FILEB in TO directory.

```
COPY FILEA1 FILEA2, FILEB1 FILEB2, FILEC1 FILEC2
```

copies FILEA1, FILEB1, and FILEC1 in the FROM directory to FILEA2, FILEB2, and FILEC2 in the TO directory.

```
COPY (Ø)
```

In this case, the FROM directory and TO directory must each be segment directories. COPY copies the file at position (0) of the FROM directory to position (0) of the TO directory. There are no access rights associated with these files, so PRIMOS checks instead the access rights of the directory. A user cannot set the FROM and TO directories if they are segment directories without access rights to them. No spaces are allowed in the name (0).

```
COPY (Ø)  (1)
```

copies the file at position (0) of the FROM directory to position (1) of the TO directory, both of which are segment directories.

▶ COPYDAM filea [fileb] [,filec filed] ...

The function is the same as COPY, but COPYDAM sets file type of **fileb** and **filed** to DAM, instead of copying the type of **filea** and **filec.**

▶ COPYSAM filea [fileb] [,filec [filed]]...

The function is the same as COPY but COPYSAM also sets file type of **fileb** and **filed** to SAM, instead of copying the type of **filea** and **filec**.

▶ CREATE ufdname [owner-password [nonowner-password]]

CREATE creates a **UFD** in the TO directory and assigns any **owner** and **nonowner passwords** specified. A UFD of the same name cannot already exist in the TO directory. If a password is not specified, it is set to six spaces (null). If a password is specified, longer than six characters, only the first six characters are used. The access rights of the new UFD are the default access rights set by PRIMOS. Refer to the CREATE command in Section 3.

▶ DELETE filea [fileb]...

The DELETE command deletes **filea** and optionally **fileb** from the FROM directory. **filea** and **fileb** cannot be directories. The user must have read, write, and delete access rights to each file specified. If **filea** and **fileb** are in a segment directory, read, write, and delete access rights are required for the FROM directory.

Examples:

```
DELETE FILEA

DELETE FILEA FILEB FILEC FILED

DELETE (1)  (2)  (185)  (186)
```

▶ FORCE {ON / OFF}

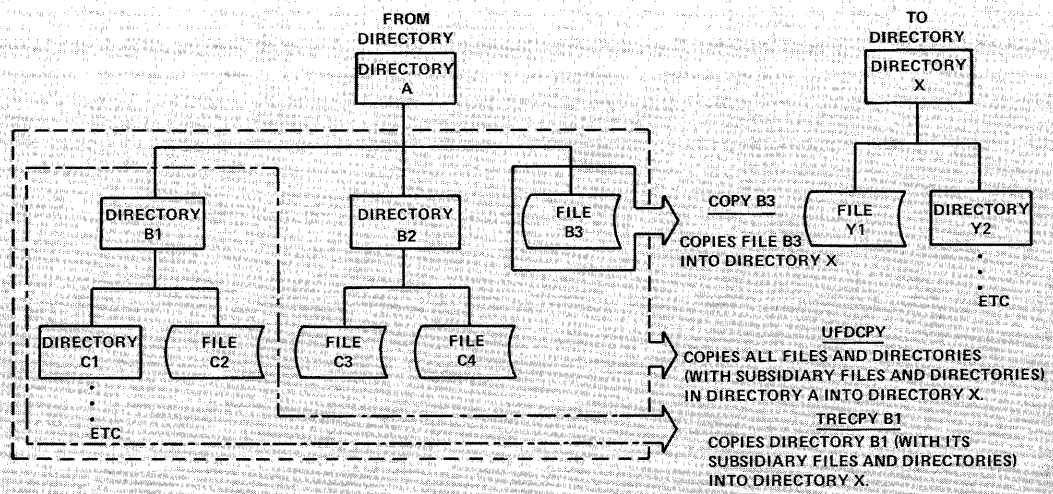
As noted previously, LISTF, LISTSAVE, SCAN, UFDCPY, and TRECPY will not force read access rights on any files or subdirectories within the FROM directory. This restriction not only prevents the updating of the date-and-time stamp of files to be copied; it also permits these commands to work on write-protected disks. The price of this capability is that all files to be listed or copied must have read access. To override this restriction, the command **FORCE ON** must be specified. This command causes read access rights to be forced, but also causes LISTF, LISTSAVE, SCAN, UFDCPY, and TRECPY to fail on write-protected disks. The option remains in force until the command **FORCE OFF** is specified. UFDCPY never forces rights on the primary level of either the FROM or TO directory.

The user must have owner rights in the UFD to give the FORCE command.

▶ FROM pathname

where **pathname** is of the form:

<ldisk>
<packname> directory [password-1] [ldisk] > directory [password-2]...
<*>



COPYING

COPY B3	COPIES FILE B3 INTO DIRECTORY X
TRECPY B1	COPIES DIRECTORY B1 (WITH ITS SUBSIDIARY FILES AND DIRECTORIES) INTO DIRECTORY X
UFDCPY	COPIES ALL FILES AND DIRECTORIES (WITH SUBSIDIARY FILES AND DIRECTORIES) IN DIRECTORY A INTO DIRECTORY X

DELETING

DELETE B3	DELETES FILE B3 FROM DIRECTORY A
TREDEL B1	DELETES DIRECTORY B1 (WITH ITS SUBSIDIARY FILES AND DIRECTORIES) FROM DIRECTORY A
UFDDEL	DELETES ALL FILES AND DIRECTORIES (WITH SUBSIDIARY FILES AND DIRECTORIES FROM DIRECTORY A. DIRECTORY A STILL EXISTS WITH .NULL.CONTENTS

PROTECTING

PROTEC {B1 / B2 / B3} [OWNER[NONOWNER]]	SETS OWNER AND NON-OWNER ACCESS RIGHTS FOR THE FILE OR DIRECTORY SPECIFIED ONLY.
TREPRO B1 [OWNER[NONOWNER]]	SETS OWNER AND NON-OWNER ACCESS RIGHTS FOR DIRECTORY B1 AND ALL SUBSIDIARY DIRECTORIES (C1, ETC.)
UFDPRO [OWNER[NONOWNER[LEVELS]]]	SETS OWNER AND NON-OWNER ACCESS RIGHTS FOR B1,B2,AND B3. ALSO SETS RIGHTS FOR SUBSIDIARY FILES AND DIRECTORIES IN B1 AND B2 TO THE SPECIFIED NUMBER OF LEVELS. THE DEFAULT VALUE FOR LEVELS IS 1.

FOR ALL PROTECTION COMMANDS THE DEFAULT ACCESS RIGHTS ASSIGNED ARE:

OWNER 1 (READ ONLY) NON-OWNER 0 (NO RIGHTS)

THIS IS NOT THE SAME AS THE PRIMOS DEFAULTS WHICH ARE 7 AND 0

Figure 4-3. FUTIL: COPYing, DELETing and PROTECTion Commands

FROM defines the FROM directory in which files are to be searched for the commands COPY, COPYSAM, COPYDAM, DELETE, LISTF, LISTSA, SCAN, CLEAN, PROTECT, TREPRO, UFDPRO, TRECPY, TREDEL, UFDCPY, and UFDDEL. The FROM directory is defined from the **pathname** parameter (see format above). The **pathname** may contain up to 10 directories that can be segment directories as well as User File Directories. If segment directories are specified, the user must have read access rights to them. If any error is encountered, the FROM directory is set to the home UFD (*). The first directory in the **pathname** may be *, which refers to the home UFD. The default FROM directory is the home UFD. Use of FROM never changes the home UFD. If the FROM name is a relative pathname (i.e., begins with *), any subsequent ATTACH commands that change the home UFD will reset the FROM name to *.

Examples:

```
FROM  <Ø>BEECH
```

Set FROM directory to BEECH on logical disk 0. BEECH must be in the MFD on logical disk 0 and have a blank password.

```
FROM  BEECH  ABC
```

Search the MFD on all started disks for BEECH in logical disk order 0 to n. Set the FROM directory to the first directory encountered named BEECH. One of the passwords of BEECH must be ABC.

```
FROM  <TDISK>BEECH>SUB1>SUB2
```

Set the FROM directory to SUB2. SUB2 must be a directory in SUB1; SUB1 must be a directory in BEECH; and BEECH must be a directory in the MFD on a disk with packname TSDISK. The directories BEECH, SUB1, and SUB2 must also have a blank password.

▶ FROM*

Set the FROM directory to the home UFD. The home UFD is normally the last UFD the user has logged into, or attached to with either the ATTACH or FUTIL ATTACH command. If one were logged into BEECH, the above command sets the FROM directory effectively to BEECH. This command does not have to be given again if the user changes the home UFD. Furthermore, this command does not have to be given at all unless the FROM directory has been made something other than the home UFD, since the home UFD is the default.

▶ LISTF [level] [FIRST] [LSTFIL] [PROTEC] [SIZE] [RWLOCK] [TYPE] [DATE] [PASSWDS]

Lists the FROM directory, the TO directory treename, and all files and directory trees in the FROM directory at the terminal. LISTF optionally follows each filename by its protection attributes: size in disk records (modulo 440 words); file type; data/time modified; and, on directories, owner and nonowner passwords. The user must give the owner password in specifying the FROM directory. If the LSTFIL option is given, the list of files is sent to a file named LSTFIL in the home UFD instead of to the terminal. At a later time, a user may print that file on a line printer. **level** is a number specifying the lowest level in the FROM directory tree structure to be listed. The order of these options is not significant. (See Figure 4-2.) The following list describes the output.

Level	Output
0	The FROM directory name
1	The FROM directory and all files and directories within it (level 1 directories)
2	All output at level 1 and all files and directories in level 1 directories

If the level is omitted, the default is 1.

The protection attribute of each file is printed as: Owner-Key Nonowner-Key. These keys have the following meanings:

Octal Value	Key	Meaning
0	NIL	No access allowed
1	R	Read access only
2	W	Write access only
3	RW	Read and write access
4	D	Delete/truncate only
5	RD	Delete/truncate and read
6	WD	Delete/truncate and write
7	RWD	All access allowed

Where the components of the **key** values have the following meaning:

R	Read rights
W	Write rights
D	Delete and truncate rights
NIL	No rights

Any combination of owner and non-owner rights is possible.

Examples:

```
O:RWD  N:NIL
```

means the owner has read, write, delete and truncate rights; nonowners have no rights.

```
O:RWD  N:R
```

means the owner has all rights; nonowners have read rights.

The possible file types are:

SAM	For SAM file
DAM	For DAM file
SEGSAM	For SAM segment directory
SEGDAM	For DAM segment directory
UFD	For User File Directory

The DTM of a file or directory is printed as:

```
15:31:22 MON Ø8-NOV-1976
```

where 15:31:22 is 15 hours past local midnight (3 PM), 31 minutes, 22 seconds. The printed day of the week will be correct for all dates between 1 January 1972 and 31 December 2071. If the date is unreasonable (e.g., when SETTIM -000000 -0000 is typed at the supervisor terminal) the DTM is not printed.

All dates are considered to be reasonable as long as the month is between 1 and 12. Note that the day of the week will be correct for dates such as 32 December and 0 April because they will be considered as 1 January and 31 March, respectively.

The passwords on subdirectories are printed as (owner, nonowner). Non-printing characters are suppressed rather than replaced by blanks or printed on the user terminal although they will be sent to the output file (LSTFIL) (not to comoutput files, however). The default passwords on a UFD are printed as (,).

The FIRST option specifies that all files with names not beginning with * (the usual convention for run files) and B_(the usual convention for PMA, FTN and COBOL object files) are to have their first lines printed. If the file is not an ASCII file and the name does not begin with * or B_, the comment (NO FIRST LINE) will be printed. First lines are preceded by a colon and a space and are placed on the same line as the file and its options, if they fit.

LISTF traverses the file structure (as shown by the meandering line in Figure 4-4) generating printed messages in sequence (as shown in the circles adjoining the meandering line).

If the optional parameter RWLOCK is specified, the FUTIL LISTF command also prints the value of per-file read-write lock settings for files. FUTIL does not print this information for UFD names, since it is not significant in that case. The value of the per-file read-write lock is printed as follows:

RWLOCK Value	Meaning
SYS	Use the system read-write lock settings.
W/NR	Allows N readers or one writer to have the associated file open.
1WNR	Allows N readers and one writer to have the associated file open.
NWNR	Allows N readers and N writers to have the associated file open.

When LISTF is used to produce a list of the sample file configuration shown in Figure 4-4, the output level is set to 3 with the SIZE option.

▶ LISTSAVE filename [level] [PROTEC] [SIZE] [TYPE] [DATE] [RWLOCK] [PASSWDS] [FIRST]

The LISTSAVE command is identical in function to the LISTF command with the LSTFIL option specified, except the output listing file is named with the name specified by **filename** rather than LSTFIL, and the LSTFIL option is redundant.

▶ PROTECT filename [owner-access [nonowner-access]]

PROTECT will protect **filename** in the FROM directory with the **owner** and **nonowner** protection attributes specified. Refer to LISTF for a description of protection attributes. If the nonowner rights are omitted, they are set to 0. If the owner rights are omitted, they are set to 1 (read only). **Filename** can be a file, a UFD, or a segment directory. If it is a UFD, the files and subdirectories within it will not be protected.

▶ QUIT

The QUIT command of FUTIL returns to PRIMOS command mode.

▶ SCAN filename [level] [PROTEC] [SIZE] [TYPE] DATE] [PASSWDS] [LSTFIL] [FIRST] [RWLOCK]

The SCAN command is used to search the FROM directory tree for the occurrence of all files, sub-UFDs, and segment directories that are named with the name specified by **filename.**

If the **level** specified by the argument level is 1 (the default), only the **filename** followed by the information specified by the optional arguments is printed. If the level specified by **level** is greater than 1, the **pathname** (treename) to the file or directory, starting from the FROM directory, is printed. In addition, the information specified by any optional arguments may be printed after the treename. For example, with the sample tree structure shown in Figure 4-4, the command:

```
SCAN FILEB 1Ø S F
```

will print the following information:

```
FROM-DIR = MFD
TO-DIR   = *

DIRECTORY PATH = MFD> UFD> SUFD12

   FILEB              1   (NO FIRST LINE)
```

FILEB lacks a first line because it was empty. The name was indented three spaces because it is a sub-UFD that is a third level in a tree subordinate to the MFD.

▶ SRWLOC filename number

SRWLOC sets the per-file read-write lock for the file specified by **filename**. The parameter **number** is an octal number that is the read-write lock setting. The read-write lock is interpreted as modulo 4; 0 means use the system read-write lock, 1 means allow multiple readers or one writer, 2 means allow multiple readers and one writer, 3 means allow multiple readers and multiple writers.

▶ TO pathname

TO defines the TO directory in which files are searched for the commands CREATE, COPY, COPYSAM, COPYDAM, TRECPY, and UFDCPY. The TO directory is defined from the **pathname** parameter, which has a format similar to the directory pathname specified for the FROM command. The **pathname** may contain at most 10 directories that may be segment directories as well as UFDs. If segment directories are specified, the user must have read and write access to them. The first directory in the **pathname** may be the home UFD (*). The default TO directory is the home UFD. If any error is encountered, the TO directory is set to the home UFD (*).

Note

The TO command never changes the home UFD. If the TO name is a relative pathname (i.e., begins with *), any subsequent ATTACH commands that change the home UFD will reset the TO name to *.

▶ TRECPY dira [dirb] [,dirc [dird]]

TRECPY is used to copy directory trees. A directory tree consists of all files and subdirectories that have their root in that directory.

The command TRECPY copies the directory tree specified by directory **dira** to directory **dirb**, and optionally **dirc** to **dird**. **dirb** and **dird** must not exist prior to the TRECPY command. If **dirb** is omitted, **dira** is taken as the name of the directory to be copied to. **dira** and **dirc** must be in the FROM directory; **dirb** and **dird** are created in the TO directory. Read access rights are required for **dira** and **dirc** and all files or subdirectories within them. The restriction on subdirectories can be overridden by use of the FORCE command (described in this section).

The directories **dirb** and **dird** are created with the same directory type and passwords as **dira** and **dirc**, and with default access rights. Also, the per-file read-write lock setting is copied by TRECPY. The names, access rights, and passwords of all files and subdirectories are also copied.

Example:

```
FROM MFD
TO   MFD
TRECPY   BEECH   LAUREL
```

copies the directory tree specified by BEECH in the MFD to a new directory, LAUREL, in the MFD.

▶ TREDEL dira [dirb]

The TREDEL command deletes the directory tree specified by directory **dira** and optionally deletes **dirb** from the FROM directory. **dira** and **dirb** must be directories. The user must have read, write, and delete rights to the **dira** and **dirb**; however, read, write, and delete rights are not required for files and subdirectories nested with the **dira** and **dirb**.

Note

The PRIMOS operating system DELETE command will not delete a directory if it is not empty.

▶ TREPRO directory [owner-access [nonowner-access]]

The TREPRO command is the same as PROTECT, except that directory is a UFD or segment directory in the FROM directory and it and all files under it (UFDs only) are protected with the specified access rights. The default access rights are <1 0>.

▶ TRESRW directory number

The TRESRW command sets the per-file read-write locks for all files in the sub-tree beginning with the directory (segment directory or UFD) specified by **directory**. The parameter **number** is the read-write lock settings, which are discussed in the description of the FUTIL command SRWLOC.

▶ UFDCPY

This command, UFDCPY, copies all files and directory trees from the FROM directory to the TO directory. The user must have owner rights in the FROM directory. Furthermore, all files and directories in the FROM directory must have read access rights. This restriction on subdirectories can be overridden with the FORCE command (described in this section).

Files already existing in the TO directory with names identical to those in the FROM directory are replaced. The user must have read, write, and delete access rights to files that are to be replaced.

Segment directories already existing in the TO directory with names identical to those in the FROM directory are not allowed and will not be copied. Files and directories created in the TO directory will have the same file type and access rights as the old files. If a file or UFD in the TO directory has the the same name as a file or UFD in the FROM directory, the access rights must permit read, write, and truncate /delete. When the copy is finished, the new file will have the same protection attributes as the corresponding file in the FROM directory. The names, access rights, per-file read-write lock settings, and passwords of all files and subdirectories within directory trees being copied are also copied. Other existing files and directories in the TO directory are not affected. UFDCPY is effectively a merge of two directories including merging sub-UFDs. Both the FROM and the TO directory must be user-file directories.

Example:

```
FROM   BEECH
TO     LAUREL
UFDCPY
```

copies all files and directories from BEECH to LAUREL (BEECH and LAUREL are in the MFD). Note that, unlike the example for TRECPY, the user has not specified the MFD as the FROM directory; therefore, he does not need to know the MFD password. In the example, LAUREL exists prior to the UFDCPY. With the TRECPY example, LAUREL does not previously exist.

▶ UFDDEL

The UFDDEL command deletes all files and directory trees (specified by directories) within the FROM directory. The user must give the owner password in the FROM command and have read, write, and delete access to all files and directories within the FROM directory. These rights are not required for files and subdirectories nested within the directories in the FROM directory.

Note

Read and write access rights to a sub-UFD are sufficient to delete the contents of that directory, but not to delete the directory itself.

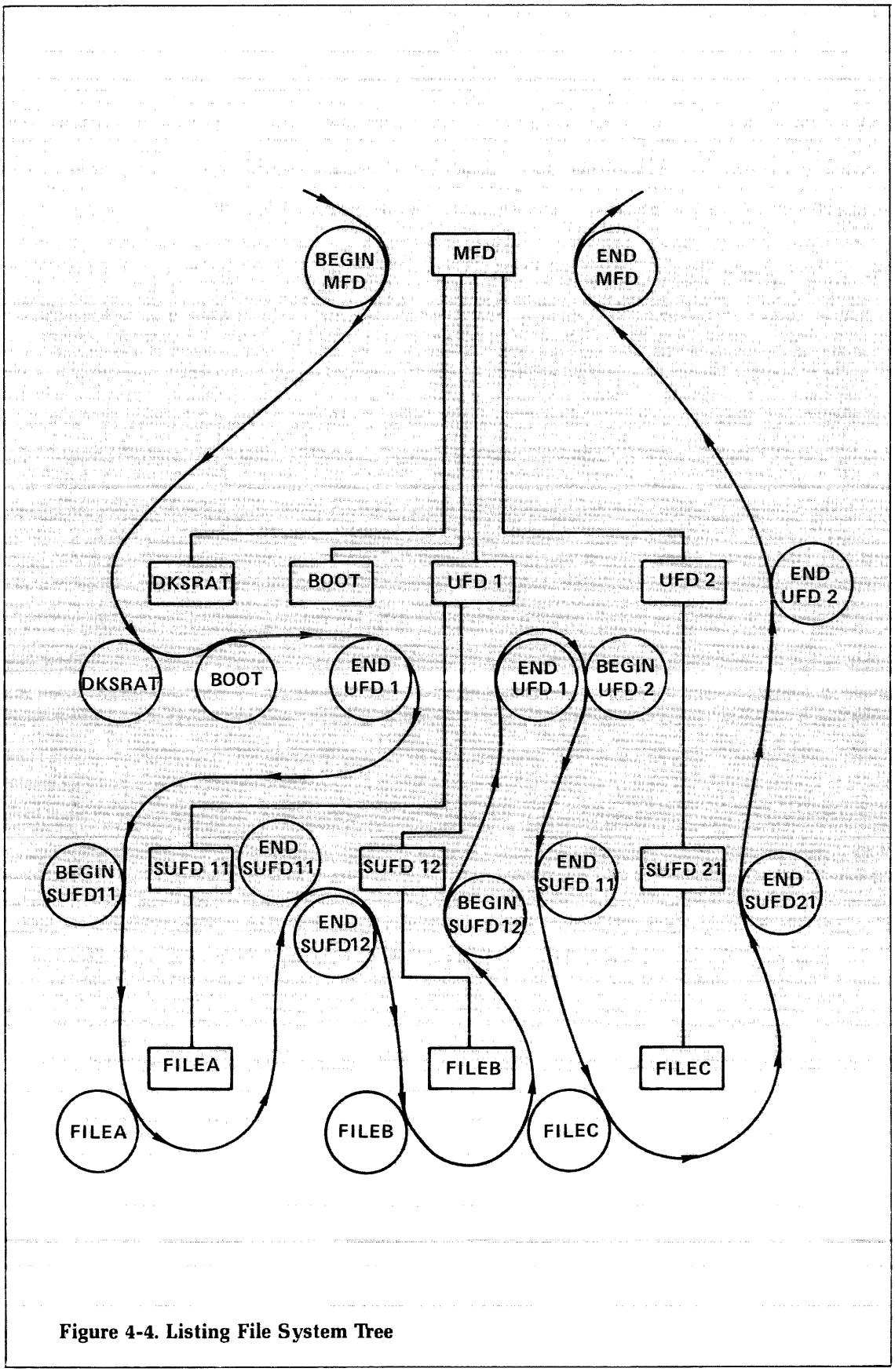


Figure 4-4. Listing File System Tree

▶ UFDPRO [owner-access [nonowner-access [levels]]]

The UFDPRO command is used to protect all files and directories within the FROM directory with the specified rights, going down sub-UFD trees the specified number of **levels**. The default rights are <1 0> and the default level is 1. Thus, in the example structure of LISTF, SCAN, and CLEAN, the command: UFDPRO will protect the files DSKRAT and BOOT and the UFDs UFD1 and UFD2 with access rights <1 0> and will not change the rights of any of the subdirectory UFDs or files. The command: UFDPRO 1 0 10 will protect all files and directories within MFD. Both the owner and nonowner access rights must be specified in order to specify the number of levels.

▶ UFDSRW number [levels]

The UFDSRW command sets the per-file read-write locks for levels of files in the FROM directory. The parameter **number** is the read-write lock setting which is discussed in the description of the FUTIL command SRWLOC. (Default for **levels** is 1.)

RESTRICTIONS

FUTIL cannot process UFD filenames that contain the characters ( ) <> [ ] or , . Avoid using filenames containing these characters. (Segment directory filenames use parentheses around the numerical file specifier.) In using FUTIL under PRIMOS, certain operations may interfere with the work of other users. For example a UFDCPY command to copy all files from a UFD currently used by another logged-in user may fail. If any file in that directory is open for writing by that user, UFDCPY will encounter the error "file in use", and will skip the file. If the user attempts to open one of his files for writing while UFDCPY is running, the user may encounter that error. The FUTIL LISTF and TRECPY commands cause the same interaction problems. Other FUTIL commands such as COPY and DELETE can also interfere with the other user, but the problem is not as serious as only one file is potentially involved in a conflict. To prevent the conflicts, users working together and involved in operations using each other's directory should coordinate their activities. If two users consistently use the same UFD at the same time, they should avoid the FUTIL LISTF command, and use the system LISTF command instead.

FUTIL operations when using the MFD should be done carefully. Never give the command TREDEL MFD, since the command will delete every file on the disk except the MFD, disk record availability table, BOOT, and BADSPT. A LISTF or UFDCPY of the MFD should be done only if one is sure no other user is using any files or directories on that disk. A UFDCPY of the MFD to the MFD of another disk has the effect of merging the contents of two disks onto one disk. A user should be sure there is enough room on the TO disk before attempting this operation or it will abort. Recall also that the names of segment directories on the two disks must not conflict. Files of the same name will be overwritten and UFDs of the same name will be merged. To avoid the name conflict, it may be desirable to UFDCPY the MFD of one disk into a user-file-directory on another disk. Each directory originally on the FROM disk becomes a subdirectory in that UFD on the TO disk. A UFDCPY of an MFD does not copy the DSKRAT, MFD, BOOT or BADSPT to the TO directory. If a user wishes to copy BOOT to the TO directory, use the COPY command. Never copy the DSKRAT or the BADSPT file from one MFD to another.

The effect of a UFDCPY from the MFD of a disk in use to the MFD of a newly formatted disk is to reorganize the disk files so that all files are compacted, that is, have their records close to each other on the new disk. After such a compaction, the access time to existing files on the new disk is less than the access time on the old disk. Furthermore, new files tend to be compact since all free disk records are also compacted. The use of such compacted disks should improve the performance of all PRIMOS systems.

Users should not abort copying or deleting operations under PRIMOS II, but should allow them to run to completion. Aborting a copy or delete operation may cause a pointer mismatch or bad file structure or a directory with a partial entry. PRIMOS will not run correctly with a directory with a partial entry. FIXRAT should be run immediately if these conditions are encountered (see

the **System Administrator's Guide**). Under PRIMOS, interruption of FUTIL with CONTROL-P will never result in a bad file structure.

FUTIL ERROR MESSAGES

The following are error messages generated by FUTIL. In many cases, FUTIL types error messages generated by PRIMOS and retains control, so users should be generally familiar with operating system error messages. The list given here includes those messages that may be encountered by FUTIL. Most messages are preceded by a filename identifying the file causing the error. Some of the error messages have the format:

```
reason for error
FILE =                    filename
```

In all cases except ALL FILE UNITS IN USE or DISK FULL on copies, FUTIL will continue with the operation, reporting all errors as it goes until the operation is complete.

? CAN'T ABBREVIATE XXXXX COMMAND

The command given is an abbreviation for a command considered too dangerous to abbreviate. Give the command in full if you wish to use it.

? OPERATION ILLEGAL INSIDE SEGDIRS

A PROTECT- or SWRLOC-class command was given while the FROM-dir was inside a segment directory.

? UNKNOWN COMMAND- XXXXXX

An unrecognizable command was given.

ALL FILE UNITS IN USE. I NEED AT LEAST 6 (FUTIL)

FUTIL no longer closes file units when it is invoked. Instead, it allocates its own units from any free units available. FUTIL needs between 6 and 14 free units.

ALREADY EXISTS or **SEG DIR ALREADY EXISTS**

An attempt has been made to TRECPY to or CREATE a UFD or segment directory that already exists; or UFDCPY has attempted to copy a segment directory which already exists. If you intend to do the operation, the UFD or segment directory in the TO directory must first be deleted.

file ALREADY EXISTS WITH WRONG FILE TYPE

Indicates an attempt was made to copy a file into an existing segment directory or to TRECPY a segment directory into an existing file.

ALREADY OPEN

Indicates an attempt to UFDCPY a directory to itself, or an attempt to copy a file to itself, or an attempt to copy a directory to a subdirectory within itself.

BAD NAME

A segment directory filename was given to a command which expected a UFD filename or vice versa. The type of filename must match the type of directory the file is contained in.

BAD PASSWORD

An incorrect password has been given in a FROM, TO, or ATTACH COMMAND. PRIMOS will not allow FUTIL to maintain control in case of a bad password so the FUTIL command must be given to restart FUTIL after the user has attached to his directory.

Since FUTIL accepts lowercase input, it is conceivable that while FUTIL correctly interprets lowercase for commands, filenames, and options, some user may forget that all passwords are still interpreted literally. For example, a UFD with a password of ABC can only be attached with owner-rights if the password is entered literally as uppercase ABC.

BAD STRUCTURE

Indicates any of various conditions in which the implied or explicitly specified structure is illegal. For example, an attempt to specify a UFD under a segment directory will cause this error.

BAD SYNTAX

The command line processed by FUTIL is incorrect.

CANNOT ATTACH

An attempt is made to UFDCPY a directory in which a sub-UFD has the same name as a file or segment directory on the TO side. The FROM side UFD is skipped.

CANNOT ATTACH TO SEGDIR

The last directory in the directory pathname to an ATTACH command is a segment directory. It must be a UFD, as ATTACH sets the home-UFD to the last directory in the path.

CANNOT COPY FILE TO DIRECTORY

On UFDCPY, indicates a file on the FROM side has the same name as a directory on the TO side.

CANNOT DELETE MFD

User has given the UFDDEL command while attached to the MFD. This is not allowed.

DISK ERROR

Same as **UNRECOVERED ERROR.**

DISK FULL

The disk has become full before FUTIL has finished a copy operation. For operations involving many files, some files are not copied, creating only partially copied directories which may be of limited use. It is suggested that the user delete such a structure immediately to prevent confusion as to what has been copied.

END OF FILE

User has attempted to reference a nonexistent file beyond the end of a segment directory.

IN USE

Indicates a FUTIL attempt to process a file in use by some other user. It may also indicate an attempt to copy a directory to a subdirectory within itself.

INSUFFICIENT ACCESS RIGHTS

User has attempted an operation on a file which violates the file access rights assigned to that file. These rights may be changed by the PROTECT command, if the user has given the owner password on ATTACH.

NO UFD ATTACHED

Self-explanatory. Often a result of misspelled/cased password.

NOT A DIRECTORY

User has given a directory-treename which includes a regular file.

NOT FOUND

Self-explanatory. Often due to typographical errors.

NOT FOUND IN SEG-DIR

User has attempted to reference a file in a segment directory with an entry of 0, which indicates file does not exist or the user has attempted to reference a file past the end of the segment directory.

POINTER MISMATCH

Indicates a bad file structure. Running FIXRAT is in order.

STRUCTURE TOO DEEP

Directories may be nested to a depth of 100 levels. User has attempted to exceed this limit.

TOO MANY NAMES

A FROM, TO or ATTACH treename was specified with more than 10 names.

UFD FULL

On a UFDCPY merge or a UFDCPY or TRECPY from a new partition to an old partition, the TO directory or a subdirectory has become full. FUTIL will report the error and then pop-up a level and continue as if the UFD had not become full.

UNRECOVERED ERROR

Indicates either an attempt to write to a write-protected disk, an actual disk error, or a FUTIL attempt to process a bad file structure. Running FIXRAT is in order if the disk was not write-protected.

WRONG FILE TYPE

An attempt was made to DELETE or COPY a directory, or to TREDEL, TRECPY or TREPRO a file.

A

RVEC PARAMETERS

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 LOADer's or SEG's SAVE command, when the program was stored on disk.

Each parameter is a 16-bit processor word, represented by up to six octal digits.

Parameter	Memory Location	Definition
SA	—	Starting Address (first memory word used by program)
EA	—	Ending Address (last memory word 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, 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
```

sets the value of the RVEC parameter, B (i.e., skip two octal parameters and set the third to '1000).

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.

RESTOR returns a program from disk to memory and loads the SAVE parameters into RVEC in preparation for a START command.

RESUME combines the functions of RESTOR and START.

PM lists the current values of the RVEC parameters.

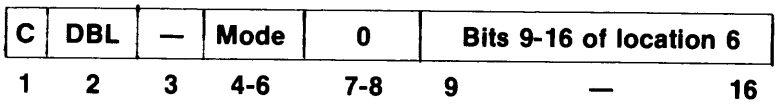
External commands have RVEC parameters that can be modified at the time the command is started (e.g., PMA filename 1/740). Providing RVEC parameters to a command that does not need them will cause unpredictable results.

Keys

The item **keys**, when specified among the RVEC parameters, refers to the processor status keys handled by the INK and OTK instructions. (Refer to the **Reference Guide, System Architecture.**) These are represented by a single 16-bit word in one of the following formats. (S-mode and R-mode programs use the first format; V-mode and I-mode programs use the second).

Keys (SR)

Process status information is available in a word called the keys, which can be read or set by the program. Its format is as follows:



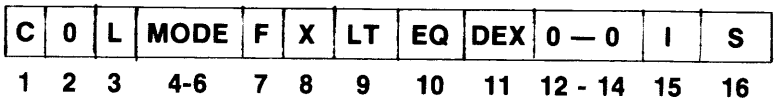
C (Bit 1)	Set by arithmetic error conditions
DBL (Bit 2)	0 - Single Precision, 1 - Double Precision.
MODE (Bits 4-6)	The current addressing mode as follows:

000	16S
001	32S
011	32R
010	64R
110	64V
100	32I

C-bit (SR): Bit 1 in the keys. Set by arithmetic error conditions and shifts (Bit 1).

Keys (VI)

Process status information is available in a 16-bit register known as the keys. It may be referenced by the LPSW, TKA, and TAK instructions.



C (Bit 1)	C-Bit
L (Bit 3)	L-Bit
MODE (Bits 4-6)	Addressing Mode:

000	16S
001	32S
011	32R
010	64R
110	64V
100	32I

F (Bit 7)	Floating point exception disable: 0 take fault 1 set C-bit
X (Bit 8)	Integer exception enable 0 set C-bit 1 take fault
LT (Bit 9)	Condition code bits:
EQ (Bit 10)	LT set if result is negative EQ set if result is zero
DEX (Bit 11)	Decimal exception enable 0 set C-bit 1 take fault
I (Bit 15)	In dispatcher — set/cleared only by process exchange
S (Bit 16)	Save done — set/cleared only by process exchange

C-bit (VI): Set by error conditions in arithmetic operations and by shifts.

L-bit (VI): 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. It is valuable for simulating multiple-precision operations and for performing unsigned comparisons following a CAS or a SUB.

Condition code bits (VI): The two condition-code bits are designated "EQ" and "LT". EQ is set if, and only if, the result is zero. If overflow occurs, EQ reflects the state of the result after truncation rather than before. LT reflects the extended sign of the result (before truncation, if overflow), and is set if the result is negative.

B

PAPER TAPE PUNCH AND COPY UTILITIES

This appendix describes the use of the paper tape utilities MDL and PTCPY on a Prime Computer that does not have mass storage — for example, a Prime 200 used in an on-line control application and using paper tape as the data entry medium.

MDL and PTCPY can also be invoked from PRIMOS command level. Operating procedures are the same, once the utility takes control. The paper tape reader and/or punch must be ASSIGNed to the user.

MEMORY DUMP AND LOAD (MDL)

MDL punches paper tapes of specified sections of memory in a self-loading format that can be read by the automatic program LOAD function or an equivalent key-in loader. MDL tapes load into the same memory locations from which they are punched.

MDL first punches part of itself, a second-level bootstrap loader in 8-8 format (two tape frames per memory word image) followed by a length of leader. The memory area to be saved is then punched in a 256-word block format.

When the tape is read, the control panel LOAD function (or key-in loader) only needs to read the 8-8 format bootstrap portion of the tape. Regular program control is then transferred to the second level bootstrap, which interprets the block-format data and loads it into memory. An ASR reader is operated in full duplex so the printer is inactive while a self-loading tape is being read.

MDL punches the content of all memory locations between two specified addresses. The area need not consist of solid code or data, however; any three or more consecutive identical memory locations are compressed as follows:

Word	
1	Pattern to be repeated
2	'70 (escape character)
3	'340 (repeating word flag)
4	Number of occurrences (256 maximum.)

Versions supplied

MDL is available in three paper tape versions for the convenience of the user. One version is a self-loading tape of MDL that loads into the last sector of an 8K memory ('17000-'17777). Another self-loading 8K version of MDL, in combination with TAP, is provided under the name of TAPMDL. The combination is loaded into the top two sectors of an 8K memory (locations '16000-'17000). In addition, an object tape of MDL is provided so that each user can generate a self-loading version suitable to his particular memory configuration and program development methods. Once it is loaded into the desired area of memory, this version of MDL can punch a self-loading tape of itself. MDL occupies one sector ('777 locations) and runs in 64R addressing mode.

Loading object (relocatable) version

1. Load and initialize the Linking Loader. Specify a starting address of 'xxx000, where **xxx** is the memory area in which loading of MDL is to start. Specify 64R addressing mode.
2. Mount the MDL tape on the selected input device and prepare the device for operation.
3. Start the loader. The MDL tape should load and stop, and the loader should print LC (loading complete).

Note

Use MDL to punch a self-loading tape of itself and this operation will not need to be repeated.

Loading self-loading version

A self-loading tape of MDL or TAPMDL can be loaded from the high- or low-speed tape reader using the automatic program LOAD function or key-in loader. The TAPMDL tape uses the auto-start feature; when the tape finishes loading, TAP automatically starts and types the prompt $.

Starting

1. Set all sense switches OFF.
2. Make sure there is enough tape in tape punch to contain the memory area to be copied. Feed a few folds of leader.
3. MASTER CLEAR the CPU.

Note

Omit this step if the program to be punched uses any register file locations.

4. If the TAPMDL self-loading tape was loaded, start MDL by entering the TAP command "R 17000".

Note

Start from the panel (Step 5) if the program to be punched uses location '777. The TAP RUN command uses location '777 as a return link, so location '777 of the program will be written over before being punched.

5. If an MDL self-loading tape was loaded, turn to STOP/STEP and set the P register (Location 7) to the address specified on the MDL tape label (typically, xx000, where **xx** is the highest sector of memory). Turn to RUN and press START.
6. After MDL is started, it responds by typing the prompt:

```
SA, EA, P, K, L:
```

and waits for a string of starting parameters to be entered at the terminal keyboard.

Note

When invoked from PRIMOS, MDL starts at this point.

Entering parameters

The parameter string consists of five octal values separated by space or commas and entered by the CR or LF key. Each parameter is an octal value ranging from 0 to '177777. Leading zeroes can

be omitted. To correct a typing error, retype the parameter without a space or comma; the last six digits are retained. For a fresh start, strike any non-octal key; the request for parameters will be repeated.

The parameters are:

startloc endloc autostart keys bootloc (CR)

startloc Is the first memory location to be punched. It must be at or above '30.

endloc Is the last memory location to be punched (up to '177777).

autostart Is an **autostart** address. If specified, the CPU automatically begins execution at this location after reading the self-loading tape. If it is zero or unspecified, the CPU halts after loading tape.

keys Is a value to be inserted in the status bits associated with the INK and OTK instructions before the program begins executions. Bit assignments are defined in Figure A-1. Bits 14, 15 and 16 have special meaning for MDL.

bootloc Is the first location to be occupied by the second-level bootstrap loader, when it is read from the MDL tape during program load. If a value is not specified, the default value is 'xxx600, where **xxx** is the memory area occupied by MDL. (MDL runs in 64R addressing mode.) This parameter is required only on the first block (or tape) of a series, when bit 15 of keys is 0.

If only a starting and ending address are specified, MDL punches that memory area on the high-speed punch, automatically adds beginning- and end-of-tape records, and punches the second-level bootstrap so that it will load into the memory area occupied by MDL at the time the tape was punched. Other parameters are needed only if:

- Autostart feature is desired
- ASR punch is to be used
- Second-level loader is to be relocated
- Two or more non-contiguous memory areas are to be punched on a single tape
- Long program is to be split into two or more separate tapes (second-level loader on first tape only)

Selecting punch mechanism: Bit 16 of the keys parameter determines whether the MDL tape will be punched by the ASR or by the high-speed punch. (The default value is 0 for high-speed punch.)

Punching single block: Set bits 14 and 15 of the keys parameter to 0. The resulting tape will be punched with BOT and EOT records. This is the default value.

Punching multiple blocks on single tape: To punch several non-contiguous blocks of memory on a single tape, use the following patterns in bits 14 and 15 of the keys parameter:

Keys Bit	**14**	**15**
First block	1	0
Subsequent Blocks	1	1
Last Block	0	1

A **bootloc** should be specified when the first block is punched.

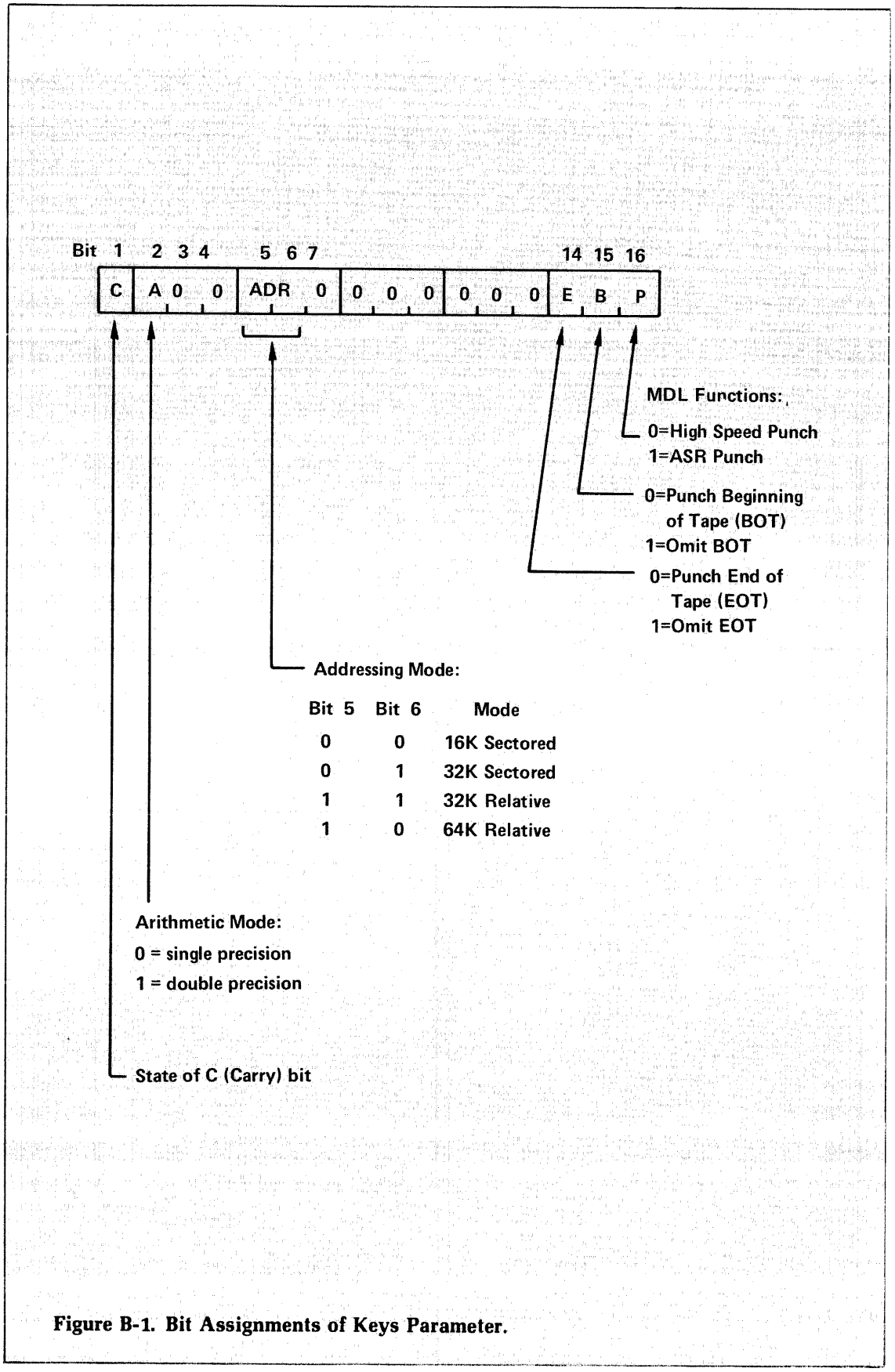


Figure B-1. Bit Assignments of Keys Parameter.

Punching multiple tapes: To break up a large program into several easily handled tapes, use the following **keys** entries:

Keys Bit	05	06	14	15
First tape	1	0	0	0
Subsequent tapes	1	0	0	1
Last tape	Any	Any	0	1

An **autostart** parameter and different addressing mode can be specified only for the last tape of the series.

A **bootloc** must be specified when the first tape is punched. Tapes prepared with this option can load only through the high-speed reader.

Address display

When the parameter string is entered, the program takes control and begins punching the self-loading tape on the selected device. If the control panel ADDRESS/DATA switch is at DATA, the indicators will display the address of each memory location being punched.

At the end of the memory block, the punch stops and the program requests another set of parameters (or returns to TAP, if present). If this is the last to be punched, remove the tape from the punch.

Aborting a punch cycle

To abort a punching operation, turn sense switch 02 ON momentarily. Punching will stop and the program will request a new set of parameters. Remove the unwanted tape and feed a new leader before restarting.

Entering parameters from panel

Parameters can be entered from the control panel rather than the terminal. Before starting MDL, load the parameters into the following CPU registers and set sense switch.

Parameter	startloc	endloc	autostart	keys	bootloc
Location	0	1	2	3	4

Then start MDL at the location specified on the tape. The parameters have the same effect as when they are entered from the terminal keyboard.

PAPER TAPE COPY (PTCPY)

PTCPY punches and verifies frame-for-frame duplicates of paper tapes of any format (source, object, or self-loading). The high-speed reader punch is required as the input and/or output device. Messages are printed on the user terminal.

Invoking PTCPY from PRIMOS

To invoke the paper tape copy from PRIMOS, first ASSIGN the paper tape reader, punch and then enter the command PTCPY. The utility responds with the prompt,

```
L, P(N), V(N), FP or FV
```

and waits for a command. The available functions are:

L Load a master paper tape into memory from the high-speed reader
P(n) Punch **n** copies of the master tape
V(n) Read **n** copies and verify them against the memory image
FP Force punch directly from master tape to copy (for tapes that are too long for the memory buffer space)
FV Force verify (for force-punched tapes)

Loading PTCPY from paper tape

PTCPY is provided as a self-loading tape that starts loading at '100 and occupies less than one sector. All of memory remaining above PTCPY is available to hold the image of the tape to be duplicated.

Available functions are:

Procedure	Action
Load PTCPY	Master clear, turn to LOAD, press START.
Load master tape into memory	Make sure all sense switches are DOWN. Set P to '100, turn to RUN, press START.
Verify tape in reader against copy in memory	Set sense switch 1 DOWN (or set P to '101), press START.
Punch Memory Contents	Set sense switch 1 DOWN (or set P to '102), press START.

Messages:

LC	Load Complete
MO	Memory Overflow
VC	Verify Complete
VE	Verify Error
PC	Punch Complete

Entering master tape

1. Mount the tape to be duplicated in the high-speed reader.
2. Set the P register (location) to '100, turn rotary mode switch to RUN and press START.

The tape will begin to load into memory. If loading is successful, PTCPY will print the LC message. If available memory runs out before the tape is completely loaded, an MO message will be printed.

Verifying memory copy

After a successful load (LC message), the copy in memory can be verified (compared frame for frame with the original tape).

1. Reposition the master tape to the beginning.
2. Set sense switch 2 UP (or set P to'101) and press START.

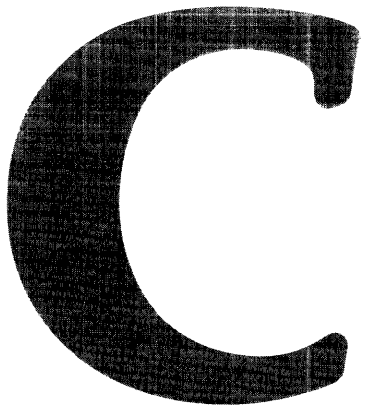
The tape will be read and compared to the copy in memory. If the memory matches the tape, the VC message will be printed; otherwise, PTCPY will print the VE message.

Punching and verifying a copy

After a tape has been read and verified as described above, duplicates can be punched:

1. Set sense switch 1 OFF (or set P to '102) and press START. A duplicate tape will be punched on the high-speed punch.
2. Remove the duplicate from the punch bin and install it in the reader.
3. Turn sense switch 1 UP (or set P to '101) and press START. The new tape will be compared with memory. A VC message means that the duplicate tape is a valid copy.
4. In case of a VE message, reposition the copy to the begining and try one more verification. If it fails again, discard the copy and punch another.

Any number of copies can be punched and verified in this way.



OLD COMMANDS

The commands in this appendix are either obsolete, or are rarely used in the detail given here.

ASRCWD is needed only by a user with a serial printer, a serial card reader, or a serial card punch.

The elaborate form of the ATTACH command became obsolete when pathnames were allowed in commands.

CX is the predecessor of the BATCH system's JOB command.

▶ ASRCWD number

ASRCWD sets a virtual control word which selects the input or output device for diverted terminal I/O. This command functions only on systems having the serial asynchronous interface used with serial printers, serial card readers, or serial card punches. After the command is given, input is taken from, and output sent to, the pair of 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:	000	User terminal
	100	User terminal
	010	Serial printer 1
	001	Serial printer 2
	000	Serial card punch

For example, to choose the serial card reader for input and the user terminal for output, select the bits:

```
xx yyyy
11 0000
```

which may be regrouped as 110 000, to become octal 60. The command is:

```
OK, ASRCWD 60
```

The virtual control word is normally set to the appropriate value by programs, and the ASRCWD command is needed only when a program exits abnormally, as by BREAK, leaving input or output diverted away from the terminal. ASRCWD 0 will correct that condition.

▶ ATTACH directory [password] [ldisk] [key]

The ATTACH command 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. They contain the name of the directory, its disk

location, and a status flag which tells whether the user is an owner or a non-owner of the directory. As an option, the user may specify that the information be recorded to redefine only the current directory, leaving the home directory information alone.

It is only possible to ATTACH to file directories, not segment directories.

Passwords

Any directory may have a pair of passwords to provide security. The owner password restricts owner-access to the files in the directory to those who know it. The nonowner password likewise restricts nonowner-access. See the PASSWD and PROTEC commands in Section 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 he supplies a password or not.

The user may give a password option of ATTACH, or as the password element of a pathname. See the discussion of pathnames in Section 2 of this guide.

Logical disk

The user may specify which logical disk is to be searched for the directory, as in the command:

ATTACH directory password ldisk

The logical disk, **ldisk**, is specified as an octal integer. Its values are:

n	Search the MFD of logical disk **n**. The LDEV column of the STATUS DISKS printout shows the logical disk number for each disk. **n** is between 0 and the number of disks allowed.
100000	Search MFDs of all disks in order of increasing **ldisk** number. (Default)
177777	Search MFD of disk to which user is currently attached.

The same actions may be accomplished when using pathnames:

ATTACH <n> pathname

ATTACH ordinary-pathname

ATTACH < * pathname

correspond exactly to ldisk values of n, 100000, and 177777 respectively.

Home key values

ATTACH ordinarily sets both the current and the home directories to the target directory. However, in the comand:

ATTACH directory password ldisk key

the value of **key** may be chosen to allow or prevent the setting of home directory to be directory. The **keys** are:

Key	Meaning
177777	Attach to a UFD in the MFD on **ldisk**; do not set as home.
0	Attach to a UFD in the MFD on **ldisk**; set as the home directory. (Default.)

1	Attach to a subdirectory in the current directory; do not set as home.
2	Attach to a subdirectory in the current directory; set as the home directory.

To specify a key without specifying ldisk, use the slash convention:

ATTACH directory password 1/key

or

ATTACH directory 1/key

In this manner, a **key** may be used with a pathname:

ATTACH pathname 1/key

▶ CX { [pathname] [-PRIORITY n] [-CPULIMIT cpu-sec] / [option] }

CX accepts jobs to be queued for subsequent execution. For example, the user may submit a lengthy FORTRAN job to compile and load, leaving the terminal free for editing another program.

To use CX, make a command input file which begins by ATTACHing to the appropriate directory, and ends with CX -E. It is wise to use a COMOUTPUT command with a CX job, lest error messages be lost. A six-character job ID code may be included on the first line of the file as a comment. This code will appear in status printouts.

Submit the job with the command

CX pathname -PRIORITY n -CPULIMIT cpu-sec

where **pathname** is the command file and **-PRIORITY n** is optional, giving a priority between 0 and 7. Higher priorities are favored over lower. The optional **-CPULIMIT cpu-sec** is the maximum number of cpu (not elapsed) seconds allowed to the job. **Cpu-sec** is between 1 and 2147483647, or is NONE (the default value). Jobs exceeding the limit are aborted.

The CX queue-managing program may allow more than one CX job to execute at a time. This means the user cannot depend upon jobs executing in the same order in which they were submitted. If execution order is important, one job can CX another at the appropriate time.

Once the job is submitted, the user can check its status with a CX command using an **option** in place of **pathname:**

Option	Function
-A	Lists entire activity file.
-Dnn	Drops the specified file (**nn**) from the queue.
-E	Must appear at end of CX command file.
-P	Lists all jobs belonging to the user.
-Q	Lists job queue.
-Snn	Lists status of specified job (**nn**).

D

DMSTK FORMAT

The DMSTK command, explained in Section 2 of this guide, traces the sequence of calls and returns by which the user's process arrived at its current state. Machine states for internal commands, condition frames, and fault frames are preserved on the user's command stack. In addition, the most recent activation of a static mode program or command is preserved on the static mode stack. DMSTK can be used to display the stack dump on the terminal or into a COMOUTPUT file. As it is an internal command, it does not overwrite the static mode stack, and so does not preclude re-entry into the faulting program.

DMSTK lists each stack frame in the following general format (For an explanation of the registers and the rings involved, see the **Reference Guide, Systems Architecture**):

```
(nn) offset: Owner= procname (LB= ownerlb).
       Called from pcl_addr; returns to return_addr.
```

The information is as follows:

Argument	Definition
nn	Frame index number of the stack frame
offset	The word number in the current stack segment where this activation's stack frame begins
procname	The name (if available) of the procedure that owns this stack frame
ownerlb	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 will return

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= fcode; fault addr= faddr.
       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
```

Argument	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, is a direct dump of the register save area (in the same format as that produced by the CPU RSAV instruction)
fcode	Fault code generated by this particular fault
faddr	Fault address generated by this particular fault

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 signalled in an inner ring and subsequently a crawlout to the current ring occurred.

If, during the trace, the stack switches to a different segment, DMSTK will print, "STACK SEGMENT IS xxxx", giving the octal segment number of the new stack segment.

Note

A called-from or returns-to value such as 0(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.

E

ASCII AND EBCDIC CHARACTER SETS

Table E-1 PRIME ASCII Character Set

^ means CONTROL key is depressed.

8-Bit Octal Code	Char		8-Bit Octal Code	Char	8-Bit Octal Code	Char	8-Bit Octal Code	Char
200	NUL		240	Sp	300	@	340	
201	SOH	^ A	241	!	301	A	341	a
202	STX	^ B	242	"	302	B	342	b
203	ETX	^ C	243	#	303	C	343	c
204	EOT	^ D	244	$	304	D	344	d
205	ENQ	^ E	245	%	305	E	345	e
206	ACK	^ F	246	&	306	F	346	f
207	BEL	^ G	247	'	307	G	347	g
210	BS	^ H	250	(	310	H	350	h
211	HT	^ I	251	)	311	I	351	i
212	LF	^ J	252	*	312	J	352	j
213	VT	^ K	253	+	313	K	353	k
214	FF	^ L	254	,	314	L	354	l
215	CR	^ M	255	-	315	M	355	m
216	SO	^ N	256	.	316	N	356	n
217	SI	^ O	257	/	317	O	357	o
220	DLE	^ P	260	0	320	P	360	p
221	DC1	^ Q	261	1	321	Q	361	q
222	DC2	^ R	262	2	322	R	362	r
223	DC3	^ S	263	3	323	S	363	s
224	DC4	^ T	264	4	324	T	364	t
225	NAK	^ U	265	5	325	U	365	u
226	SYN	^ V	266	6	326	V	366	v
227	ETB	^ W	267	7	327	W	367	w
230	CAN	^ X	270	8	330	X	370	x
231	EM	^ Y	271	9	331	Y	371	y
232	SUB	^ Z	272	:	332	Z	372	z
233	ESC		273	;	333	[	373	{
234	FS		274	<	334	\	374	|
235	GS		275	=	335	]	375	}
236	RS		276	>	336	^	376	~
237	US		277	?	337	_	377	DEL

Table E-2 EBCDIC Character Set

(bank) = bank character

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	

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
096	140	60	-	148	224	94	m
097	141	61	/	149	225	95	n
098	142	62		150	226	96	o
099	143	63		151	227	97	p
100	144	64		152	230	98	q
101	145	65		153	231	99	r
102	146	66		154	232	9A	
103	147	67		155	233	9B	
104	150	68		156	234	9C	
105	151	69		157	235	9D	
106	152	6A	¦	158	236	9E	
107	153	6B	,	159	237	9F	
108	154	6C	%	160	240	A0	
109	155	6D		161	241	A1	~
110	156	6E	>	162	242	A2	s
111	157	6F	?	163	243	A3	t
112	160	70		164	244	A4	u
113	161	71		165	245	A5	v
114	162	72		166	246	A6	w
115	163	73		167	247	A7	x
116	164	74		168	250	A8	y
117	165	75		169	251	A9	z
118	166	76		170	252	AA	
119	167	77		171	253	AB	
120	170	78		172	254	AC	
121	171	79	`	173	255	AD	
122	172	7A	:	174	256	AE	
123	173	7B	#	175	257	AF	
124	174	7C	@	176	260	B0	
125	175	7D	'	177	261	B1	
126	176	7E	=	178	262	B2	
127	177	7F	"	179	263	B3	
128	200	80		180	264	B4	
129	201	81	a	181	265	B5	
130	202	82	b	182	266	B6	
131	203	83	c	183	267	B7	
132	204	84	d	184	270	B8	
133	205	85	e	185	271	B9	
134	206	86	f	186	272	BA	
135	207	87	g	187	273	BB	
136	210	88	h	188	274	BC	
137	211	89	i	189	275	BD	
138	212	8A		190	276	BE	
139	213	8B		191	277	BF	
140	214	8C		192	300	C0	{
141	215	8D		193	301	C1	A
142	216	8E		194	302	C2	B
143	217	8F		195	303	C3	C
144	220	90		196	304	C4	D
145	221	91	j	197	305	C5	E
146	222	92	k	198	306	C6	F
147	223	93	l	199	307	C7	G

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
200	310	C8	H	228	344	E4	U
201	311	C9	I	229	345	E5	V
202	312	CA		230	346	E6	W
203	313	CB		231	347	E7	X
204	314	CC	(bank)	232	350	E8	Y
205	315	CD		233	351	E9	Z
206	316	CE	(bank)	234	352	EA	
207	317	CF		235	353	EB	
208	320	D0	}	236	354	EC	(bank)
209	321	D1	J	237	355	ED	
210	322	D2	K	238	356	EE	
211	323	D3	L	239	357	EF	
212	324	D4	M	240	360	F0	0
213	325	D5	N	241	361	F1	1
214	326	D6	O	242	362	F2	2
215	327	D7	P	243	363	F3	3
216	330	D8	Q	244	364	F4	4
217	331	D9	R	245	365	F5	5
218	332	DA		246	366	F6	6
219	333	DB		247	367	F7	7
220	334	DC		248	370	F8	8
221	335	DD		249	371	F9	9
222	336	DE		250	392	FA	|
223	337	DF		251	373	FB	
224	340	E0	\	252	374	FC	
225	341	E1		253	375	FD	
226	342	E2	S	254	376	FE	
227	343	E3	T	255	377	FF	

S

T

U

V, W